Communications on Applied Mathematics and Computation
https://doi.org/10.1007/s42967-024-00398-7

REVIEW ARTICLE

Overview Frequency Principle/Spectral Bias in Deep Learning

Zhi-Qin John Xu[1,2] · **Yaoyu Zhang**[1,2] · **Tao Luo**[1,2,3,4]

Received: 2 August 2023 / Revised: 26 February 2024 / Accepted: 4 March 2024

Abstract
Understanding deep learning is increasingly emergent as it penetrates more and more into industry and science. In recent years, a research line from Fourier analysis sheds light on this magical "black box" by showing a Frequency principle (F-Principle or spectral bias) of the training behavior of deep neural networks (DNNs)—DNNs often fit functions from low to high frequencies during the training. The F-Principle is first demonstrated by one-dimensional (1D) synthetic data followed by the verification in high-dimensional real datasets. A series of works subsequently enhance the validity of the F-Principle. This low-frequency implicit bias reveals the strength of neural networks in learning low-frequency functions as well as its deficiency in learning high-frequency functions. Such understanding inspires the design of DNN-based algorithms in practical problems, explains experimental phenomena emerging in various scenarios, and further advances the study of deep learning from the frequency perspective. Although incomplete, we provide an overview of the F-Principle and propose some open problems for future research.

Keywords Neural network · Frequency principle (F-Principle) · Deep learning · Generalization · Training · Optimization

Mathematics Subject Classification 68T01 · 68Q32 · 74S30

This paper is dedicated to the memory of Professor Zhong-Ci Shi.

✉ Zhi-Qin John Xu
xuzhiqin@sjtu.edu.cn

Yaoyu Zhang
zhyy.sjtu@sjtu.edu.cn

Tao Luo
luotao41@sjtu.edu.cn

1 Institute of Natural Sciences, MOE-LSC, Shanghai Jiao Tong University, Shanghai 200240, China

2 School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai 200240, China

3 CMA-Shanghai, Shanghai Jiao Tong University, Shanghai 200240, China

4 Shanghai Artificial Intelligence Laboratory, Shanghai 200232, China

Published online: 04 September 2024

1 Introduction

1.1 Motivation

In practice, deep learning, often realized by deep neural networks (DNNs), has achieved tremendous success in many applications, such as computer vision, speech recognition, speech translation, and natural language processing. It also has become an indispensable method for solving a variety of scientific problems. On the other hand, the DNN sometimes fails and causes critical issues in applications. In theory, the DNN remains a black box for decades. Many researchers make the analogy between the practical study of the DNN and the alchemy. Due to the booming application of DNNs, it has become an important and urgent mission to establish a better theoretical understanding of DNNs.

In recent years, the theoretical study of DNNs has flourished. Yet, we still need a clear demonstration of how these theoretical results provide key insight and guidance to practical study of DNNs. An insightful theory usually guides practice from two aspects—capability and limitation. For example, the conservation of mass in chemistry informs the fundamental limitation of chemical reactions that they cannot turn one element into another, e.g., turning bronze into gold. On the other hand, they may combine elementary substances into their compounds. These understandings are extremely valuable, with which, the study of alchemy transforms into modern chemistry. In this work, we overview the discovery and studies about the F-principle (F-Principle) of deep learning [92, 114, 115, 125], by which we obtain a basic understanding of the capability and limitation of deep learning, i.e., the difficulty in learning and achieving good generalization for high-frequency data as well as the easiness and the intrinsic preference for low-frequency data. Based on this guideline of the F-Principle, many algorithms have been developed to either employ this low-frequency bias of DNN to well fit smooth data or design special tricks/architectures to alleviate the difficulty of DNN in fitting data known to be highly oscillatory [14, 55, 73, 106]. Hopefully, with the development of the F-Principle and theories from other perspectives, the practical study of deep learning would become a real science in the near future.

The discovery of the F-Principle is made to confront the following open puzzle central for DNN theories: why over-parameterized DNNs generalize well in many problems, such as natural image classification. In 1995, Vladimir Vapnik and Larry Jackel made a bet, witnessed by Yann LeCun, that is, Larry claimed that by 2000, we would have had a theoretical understanding of why big neural nets work well (in the form of a bound similar to what we have for SVMs). Also in 1995, Leo Breiman published a reflection after refereeing papers for NIPS [11], where he raised many important questions regarding DNNs, including "why don't heavily parameterized neural networks overfit the data". In 2016, an empirical study [122] raised much attention again to this over-parameterization puzzle with systematic demonstration on modern DNN architectures and datasets. This over-parameterization puzzle contradicts the conventional generalization theory and traditional wisdom in modeling, which suggests that a model of too many parameters easily overfits the data. This is exemplified by von Neumann's famous quote "with four parameters I can fit an elephant" [28]. Establishing a good theoretical understanding of this over-parameterization puzzle has since become more and more crucial as modern DNN architecture incorporates increasingly more parameters, e.g., $\sim 10^8$ for VGG19 [102], $\sim 10^{11}$ for GPT-3 [13], which indeed achieves huge success in practice.

To address this puzzle, a notable line of works, starting from the conventional complexity-based generalization theory, attempts to propose novel norm-based complexity measures

suitable for DNNs. However, a recent empirical study shows that many norm-based complexity measures not only perform poorly, but negatively correlate with generalization, specifically, when the optimization procedure injects some stochasticity [57]. Another line of works starts from a variety of ideal models of DNNs, e.g., deep linear network [64, 95, 96], committee machine [4, 35], spin glass model [24], mean-field model [23, 82, 94, 103], and neural tangent kernel (NTK) [54, 65]. These works emphasize fully rigorous mathematical proofs and have difficulties in providing a satisfactory explanation for general DNNs [121].

The F-Principle overviewed in this paper takes a phenomenological approach to the deep learning theory, to understand complex systems, black boxes at first glance, in science and especially in physics. Taking this approach, the first difficulty we encounter is the extreme complexity of deep learning experiments in practice. For example, the MNIST dataset is a well-known simple (if not too simple) benchmark for testing a DNN. However, the learned DNN is already a very high-dimensional (784-dimensional) mapping, which is impossible to be visualized and analyzed exactly. In the face of such difficulty, an important step we take is to carefully design synthetic problems simple enough for thorough analysis and visualization of the DNN learning process, but complicated enough for reproducing interesting phenomena. We train DNNs to fit a function with 1-D input and 1-D output like $\sin(x) + \sin(5x)$ shown in Fig. 1. Luckily, a clear phenomenon emerges from the thorough visualization of the DNN training process that the DNN first captures a coarse and relatively "flat" landscape of the target function, followed by more and more oscillatory details. It seems that the training of a DNN gives priority to the flat functions, which should generalize better for flat target functions by intuition, over the oscillatory functions. By the phenomenological approach, we next quantify this phenomenon by the Fourier analysis, which is a natural tool to quantify flatness and oscillation. As shown later, by transforming the DNN output function into the frequency domain, the differences in convergence rate between flat and oscillatory components become apparent. We conclude this phenomenon of implicit low-frequency bias by the F-Principle/spectral bias [92, 114, 115, 125], i.e., DNNs often fit target functions from low to high frequencies during the training, followed by extended experimental studies for real datasets and a series of theoretical studies detailed in the main text.

In the end, as a reflection, we note that the specialness of the discovery of the F-Principle lies in our faith and insistence on performing systematic DNN experiments on simple 1-D synthetic problems, which is simple for observation and analysis but not clearly understood. From the perspective of phenomenological study, such simple cases serve as an excellent starting point, however, they are rarely considered in the experimental studies of DNNs. Some researchers even deem MNIST experiments as too simple for an empirical study without realizing that even phenomenon regarding the training of DNN on 1-D problems is not well studied. In addition, since the Fourier analysis is not naturally considered for high dimensional problems due to the curse of dimensionality, it is difficult even to think about the Fourier transform for DNNs on real datasets as done in Sect. 2 without making a direct observation of DNN learning from flat to oscillatory in 1-D problems. Therefore, by overviewing the discovery and studies of the F-Principle, we advocate for the phenomenological approach to the deep learning theory, by which systematic experimental study on simple problems should be encouraged and serve as a key step for developing the theory of deep learning.

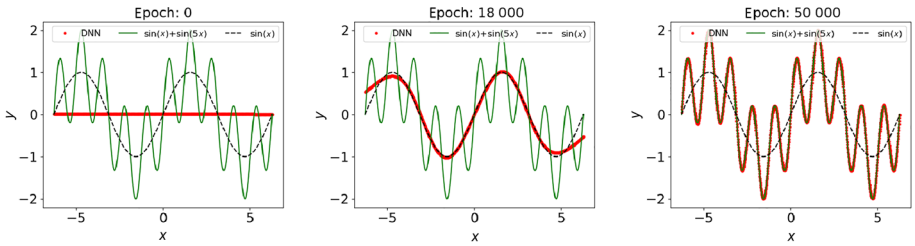


Fig. 1 The training process of a DNN. Training data are sampled from the target function $\sin(x) + \sin(5x)$. Red, green, and black curves indicate DNN output, $\sin(x)$, and $\sin(x) + \sin(5x)$, respectively

1.2 Frequency Principle

To visualize or characterize the training process in the frequency domain requires a Fourier transform of the training data. However, the Fourier transform of high-dimensional data suffers from the curse of dimensionality and the visualization of high-dimensional data is difficult. Alternatively, one can study the problem of 1-D synthetic data. A series of experiments on synthetic low-dimensional data show that the DNN training follows an F-Principle [114, 115, 125], that is,

DNNs often fit target functions from low to high frequencies during the training.

This implicit frequency bias is also called spectral bias [92] and can be robustly observed no matter how over-parameterized DNNs are. More experiments on real datasets are designed to confirm this observation [114]. It is worthy to note that the frequency used here is a response frequency characterizing how the output is affected by the input. This frequency is easy to be confused in imaging classification problems. For example, in the MNIST dataset, the frequency domain is also 784-dimensional but not 2-D, i.e., the frequency of the classification function but not the image frequency w.r.t. 2-D space.

Xu et al. [114] proposed a key mechanism of the F-Principle that the regularity of the activation function converts into the decay rate of a loss function in the frequency domain. Theoretical studies subsequently show that the F-Principle holds in a general setting with continuous samples [78] and in the regime of wide DNNs NTK regime [54]) with finite samples [77, 125, 126] or samples distributed uniformly on sphere [7, 10, 18, 119]. E et al. [33] studied the neural network from a continuous viewpoint, where neurons are treated as a discrete version of a continuous distribution of weights, and samples as a discrete version of another continuous distribution of data. The evolution of the network output during the training follows an integral equation, which would naturally lead to the training that follows the F-Principle. In addition to characterizing the training speed of DNNs, the F-Principle also implicates that DNNs prefer the low-frequency function and generalize well for low-frequency functions [77, 114, 125, 126].

The F-Principle further inspires the design of DNNs to fast learn a function with the high frequency, such as in scientific computing and image or point cloud fitting problems [15, 55, 73, 106]. In addition, the F-Principle provides a mechanism to understand many phenomena in applications and inspires a series of studies on deep learning from a frequency perspective. The study of deep learning is a highly inter-disciplinary problem. As an example, the Fourier analysis, an approach to the signal processing, is a useful tool to better understand deep learning [41]. A comprehensive understanding of deep learning remains an exciting research subject calling for more fusion of existing approaches and new methods.

2 Empirical Study of F-Principle

Before the discovery of the F-Principle, some works have suggested the learning of the DNNs may follow an order from simple to complex [3]. However, previous empirical studies focus on the real dataset, which is high dimensional. Thus, it is difficult to find a suitable quantity to characterize such intuition. In this section, we review the empirical study of the F-Principle, which first presents a clear picture from the 1-D data and then carefully designs experiments to verify the F-Principle in high-dimensional data [92, 114, 115].

2.1 Frequency Principle in Low-Dimensional Problems

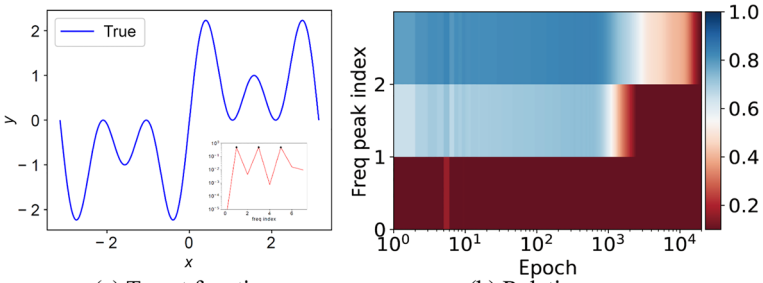
To clearly illustrate the phenomenon that follows the F-Principle, one can use 1-D synthetic data to show the relative error of different frequencies during the training of the DNN. The following shows an example from Xu et al. [114].

Training samples are drawn from a 1-D target function $f(x) = \sin(x) + \sin(3x) + \sin(5x)$ with three important frequency components and even space in $[-3.14, 3.14]$, i.e., $\{x_i, f(x_i)\}_{i=0}^{n-1}$. The discrete Fourier transform (DFT) of $f(x)$ or the DNN output (denoted by $h(x)$) is computed by $\hat{f}_k = \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)\mathrm{e}^{-\mathrm{i}2\pi ik/n}$, where k is the frequency. As shown in Fig. 2a, the target function has three important frequencies as designed (black dots at the inset in Fig. 2a). We use a network with four hidden layers consisting of 200, 200, 200, and 100 neurons, respectively. Both weights and bias are initialized from a uniform distribution on $[-\sqrt{1/m_{\text{in}}}, \sqrt{1/m_{\text{in}}}]$, where m_{in} is the number of input neurons. To examine the convergence behavior of different frequency components during the training with MSE and gradient descent, we compute the relative difference between the DNN output and the target function for the three important frequencies at each recording step, that is, $\Delta_F(k) = |\hat{h}_k - \hat{f}_k|/|\hat{f}_k|$, where $|\cdot|$ denotes the norm of a complex number. As shown in Fig. 2b, the DNN converges the first frequency peak very fast, while converging the second frequency peak much more slowly, followed by the third frequency peak.

A series of experiments are performed with relatively cheap cost on synthetic data to verify the validity of the F-Principle and eliminate some misleading factors. For example, the stochasticity and the learning rate are not important to reproduce the spectral bias phenomenon that follows the F-Principle. If one only focuses on high-dimensional data, such as the simple MNIST, it would require a much more expensive cost of computation and computer memory to examine the impact of the stochasticity and the learning rate. The study of synthetic data shows clear guidance to examine the F-Principle in the high-dimensional data. In addition, since the frequency is a quantity which the theoretical study is relatively easy to access, the F-Principle provides a theoretical direction for further study.

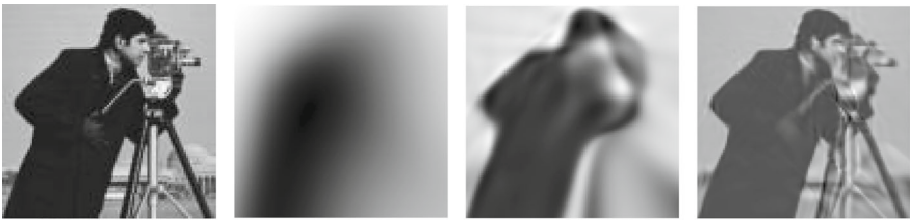
An image can be regarded as a mapping from the 2-D space coordinate to the pixel intensity. Learning this problem with a mean squared loss is a 2-D regression problem. The experiment in Fig. 3 uses a fully connected DNN to fit the camera-man image in Fig. 3a. The DNN learns from a coarse-grained image to produce one with more details as the training proceeds, shown in Figs. 3b–d. Obviously, this is also an order from low- to high-frequencies, which is similar to how biological brain remembers an image. In Xu [113], the F-Principle is examined through the quantitative characterization of the convergence of each frequency in a cross section of a 2-D regression problem.

This 2-D example also shows that utilizing DNN to restore an image may take advantage of the low-frequency preference, such as inpainting tasks, but also should be cautionary about



(a) Target function (b) Relative error

Fig. 2 1-D input. **a** $f(x)$. Inset: $|\hat{f}(k)|$. **b** $\Delta_F(k)$ of three important frequencies (indicated by black dots in the inset of (**a**)) against different training epochs. Reprinted from Xu et al. [114]



(a) True (b) Step 80 (c) Step 2 000 (d) Step 58 000

Fig. 3 F-Principle in 2-D datasets. Reprinted from Xu et al. [114]

its insufficiency in learning high-frequency structures. To overcome this insufficiency, some algorithms are developed [22, 56, 106, 110], which will be reviewed more in Sect. 4.7.

2.2 Frequency Principle in High-Dimensional Problems

To study the F-Principle in high-dimensional data, two obstacles should be overcome first: what is the frequency in high dimension and how to separate different frequencies.

The concept of "frequency" often causes confusion in image classification problems. The image (or input) frequency (NOT used in studying the F-Principle of classification problems) is the frequency of the 2-D function $I: \mathbb{R}^2 \to \mathbb{R}$ representing the intensity of an image over pixels at different locations. This frequency corresponds to the rate of change in intensity *across neighboring pixels*. For example, an image of constant intensity possesses only the zero frequency, i.e., the lowest frequency, while a sharp edge contributes to high frequencies of the image.

The frequency used in studying the F-Principle of classification problems is also called **response frequency** of a general Input-Output mapping f. For example, consider a simplified classification problem of partial MNIST data using only the data with labels 0 and 1, $f(x_1, x_2, \cdots, x_{784})$: $\mathbb{R}^{784} \to \{0, 1\}$ mapping 784-D space of pixel values to the 1-D space, where x_j is the intensity of the j-th pixel. Denote the mapping's Fourier transform as $\hat{f}(k_1, k_2, \cdots, k_{784})$. The frequency in the coordinate k_j measures the rate of change of $f(x_1, x_2, \cdots, x_{784})$ with respect to x_j, i.e., the intensity of the j-th pixel. If f possesses significant high frequencies for large k_j, then a small change of x_j in the image might induce a large change of the output (e.g., adversarial example). For a real data, the response frequency is rigorously defined via the standard nonuniform discrete Fourier transform (NUDFT).

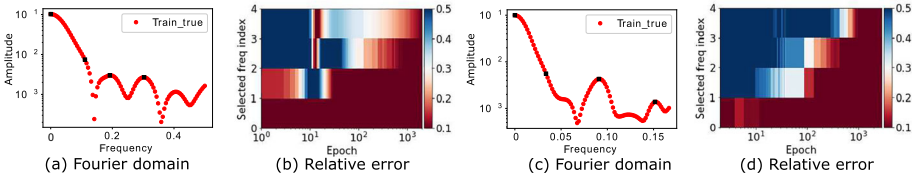


Fig. 4 Projection method. **a**, **b** are for MNIST, **c**, **d** for CIFAR10. **a**, **c** Amplitude $|\hat{y}_k|$ vs. frequency. Selected frequencies are marked by black squares. **b**, **d** $\Delta_F(k)$ versus training epochs for the selected frequencies. Reprinted from Xu et al. [114]

The difficulty of separating different frequencies is that the computation of the Fourier transform of high-dimensional data suffers from the curse of dimensionality. For example, if one evaluates two points in each dimension of the frequency space, then, the evaluation of the Fourier transform of a d-dimensional function is on 2^d points, an impossibly large number even for MNIST data with $d = 784$. Two approaches are proposed in Xu et al. [114].

2.2.1 Projection Method

One approach is to study the frequency in the 1-D frequency space. For a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=0}^{n-1}$ with $y_i \in \mathbb{R}$. The high-dimensional discrete nonuniform Fourier transform of $\{(\boldsymbol{x}_i, y_i)\}_{i=0}^{n-1}$ is $\hat{y}_{\boldsymbol{k}} = \frac{1}{n}\sum_{i=0}^{n-1} y_i \exp(-\mathrm{i}2\pi \boldsymbol{k}\cdot\boldsymbol{x}_i)$. Consider a direction of $\boldsymbol{k}$ in the Fourier space, i.e., $\boldsymbol{k} = k\boldsymbol{p}_1$, where $\boldsymbol{p}_1$ is a chosen and fixed unit vector. Then, we have $\hat{y}_k = \frac{1}{n}\sum_{i=0}^{n-1} y_i \exp\left(-\mathrm{i}2\pi(\boldsymbol{p}_1\cdot\boldsymbol{x}_j)k\right)$, which is essentially the 1-D Fourier transform of $\{(x_{\boldsymbol{p}_1,i}, y_i)\}_{i=0}^{n-1}$, where $x_{\boldsymbol{p}_1,i} = \boldsymbol{p}_1\cdot\boldsymbol{x}_i$ is the projection of $\boldsymbol{x}_i$ on the direction $\boldsymbol{p}_1$. Similarly, one can examine the relative difference between the DNN output and the target function for the selected important frequencies at each recording step. In the experiments in Xu et al. [114], $\boldsymbol{p}_1$ is chosen as the first principle component of the input space. A fully connected network and a convolutional network are used to learn MNIST and CIFAR10, respectively. As shown in Figs. 4a and c, low frequencies dominate in both real datasets. As shown in Figs. 4b and d, one can easily observe that DNNs capture low frequencies first and gradually capture higher frequencies.

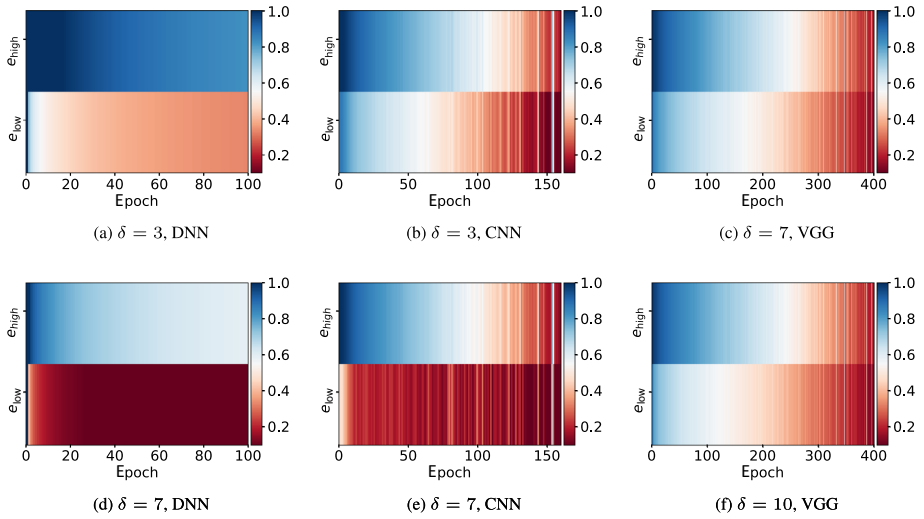
2.2.2 Filtering Method

The projection method examines the F-Principle in only several directions. To compensate the projection method, one can consider a coarse-grained filtering method which is able to unravel whether, in the radially averaged sense, low frequencies converge faster than high frequencies.

The idea of the filtering method is to use a Gaussian filter to derive the low-frequency part of the data and then examine the convergence of the low- and high-frequency parts separately. The low-frequency part can be derived by

$$\boldsymbol{y}_i^{\mathrm{low},\delta} \triangleq (\boldsymbol{y} * G^{\delta})_i, \tag{1}$$

where $*$ indicates the convolution operator, and δ is the standard deviation of the Gaussian kernel. Since the Fourier transform of a Gaussian function is still a Gaussian function but with a standard deviation $1/\delta$, $1/\delta$ can be regarded as a rough frequency width which is kept



(a) $\delta = 3$, DNN (b) $\delta = 3$, CNN (c) $\delta = 7$, VGG

(d) $\delta = 7$, DNN (e) $\delta = 7$, CNN (f) $\delta = 10$, VGG

Fig. 5 F-Principle in real datasets. e_{low} and e_{high} indicated by color against training epoch. Reprinted from Xu et al. [114]

in the low-frequency part. The high-frequency part can be derived by

$$\boldsymbol{y}_i^{\text{high},\delta} \triangleq \boldsymbol{y}_i - \boldsymbol{y}_i^{\text{low},\delta}. \tag{2}$$

Then, one can examine

$$e_{\text{low}} = \left(\frac{\sum_i |\boldsymbol{y}_i^{\text{low},\delta} - \boldsymbol{h}_i^{\text{low},\delta}|^2}{\sum_i |\boldsymbol{y}_i^{\text{low},\delta}|^2} \right)^{\frac{1}{2}}, \tag{3}$$

$$e_{\text{high}} = \left(\frac{\sum_i |\boldsymbol{y}_i^{\text{high},\delta} - \boldsymbol{h}_i^{\text{high},\delta}|^2}{\sum_i |\boldsymbol{y}_i^{\text{high},\delta}|^2} \right)^{\frac{1}{2}}, \tag{4}$$

where $\boldsymbol{h}^{\text{low},\delta}$ and $\boldsymbol{h}^{\text{high},\delta}$ are obtained from the DNN output $\boldsymbol{h}$. If $e_{\text{low}} < e_{\text{high}}$ for different δ's during the training, the F-Principle holds; otherwise, it is falsified. Note the DNN is trained as usual.

As shown in Fig. 5, the low-frequency part converges faster in the following three settings for different δ's: a tanh fully connected network for MNIST, a ReLU shallow convolutional network for CIFAR10, and a VGG16 [102] for CIFAR10.

Another approach to examine the F-Principle in high-dimensional data is to add noise to the training data and examine when the noise is captured by the network [92]. Note that this approach contaminates the training data.

3 Theoretical Study of F-Principle

An advantage of studying DNNs from the frequency perspective is that frequency can often be theoretically analyzed. This is especially important in deep learning since deep learning is often criticized as a black box due to its lack of theoretical support. In this section, we

review theories of the F-Principle for various settings. A key mechanism of the F-Principle is based on the regularity of the activation function that was first proposed in Xu [113] and was formally published by Xu et al. [114].

The theories have been developed to explore the F-Principle in an ideal setting [114], in general setting with infinite samples [78], in a continuous viewpoint [33], and in the regime of wide DNNs (Neural Tangent Kernel (NTK) regime [54]) with specific sample distributions [7, 10, 18, 117] or any finite samples [77, 125, 126].

3.1 Ideal Setting for Analyzing Activation Function

The following presents a simple case to illustrate how the F-Principle may arise. More details can be found in Xu et al. [114] and Xu [113]. The activation function we consider is

$$\sigma(x) = \tanh(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}}, \quad x \in \mathbb{R}.$$

For a DNN of one hidden layer with m nodes, 1-D input x and 1-D output

$$h(x) = \sum_{j=1}^{m} a_j \sigma(w_j x + b_j), \quad a_j, w_j, b_j \in \mathbb{R}, \tag{5}$$

where w_j, a_j, and b_j are the training parameters. In the sequel, we will also use the notation $\theta = \{\theta_{lj}\}$ with $\theta_{1j} = a_j$, $\theta_{2j} = w_j$, and $\theta_{3j} = b_j$, $j = 1, \cdots, m$. Note that the Fourier transform of $\tanh(x)$ is $\hat{\sigma}(k) = -\frac{\mathrm{i}\,\pi}{\sinh(\pi\, k/2)}$. The Fourier transform of $\sigma(w_j x + b_j)$ with $w_j, b_j \in \mathbb{R}$, $j = 1, \cdots, m$ reads as

$$\widehat{\sigma(w_j \cdot + b_j)}(k) = \frac{2\,\pi\,\mathrm{i}}{|w_j|} \exp\left(\frac{\mathrm{i} b_j k}{w_j}\right) \frac{1}{\exp(-\frac{\pi\, k}{2w_j}) - \exp(\frac{\pi\, k}{2w_j})}. \tag{6}$$

Note that the last term exponentially decays w.r.t. $|k|$. Thus,

$$\hat{h}(k) = \sum_{j=1}^{m} \frac{2\,\pi\, a_j \mathrm{i}}{|w_j|} \exp\left(\frac{\mathrm{i} b_j k}{w_j}\right) \frac{1}{\exp(-\frac{\pi\, k}{2w_j}) - \exp(\frac{\pi\, k}{2w_j})}. \tag{7}$$

Define the amplitude deviation between the DNN output and the *target function* $f(x)$ at frequency k as

$$D(k) \triangleq \hat{h}(k) - \hat{f}(k).$$

Write $D(k)$ as $D(k) = A(k)\mathrm{e}^{\mathrm{i}\phi(k)}$, where $A(k) \in [0, +\infty)$ and $\phi(k) \in \mathbb{R}$ are the amplitude and phase of $D(k)$, respectively. The loss at frequency k is $L(k) = \frac{1}{2}|D(k)|^2$, where $|\cdot|$ denotes the norm of a complex number. The total loss function is defined as $L = \int_{-\infty}^{+\infty} L(k)\mathrm{d}k$. Note that according to Parseval's theorem, this loss function in the Fourier domain is equal to the commonly used loss of the mean squared error, that is, $L = \int_{-\infty}^{+\infty} \frac{1}{2}(h(x) - f(x))^2 \mathrm{d}x$.

The decrement along any direction, say, with respect to parameter θ_{lj}, is

$$\frac{\partial L}{\partial \theta_{lj}} = \int_{-\infty}^{+\infty} \frac{\partial L(k)}{\partial \theta_{lj}} \mathrm{d}k. \tag{8}$$

The absolute contribution from frequency k to this total amount at θ_{lj} is

$$\left|\frac{\partial L(k)}{\partial \theta_{lj}}\right| \approx A(k) \exp\left(-|\pi k/2w_j|\right) F_{lj}(\theta_j, k), \tag{9}$$

where $\theta_j \triangleq \{w_j, b_j, a_j\}$, $\theta_{lj} \in \theta_j$, and $F_{lj}(\theta_j, k)$ is a function with respect to θ_j and k, which is approximate $O(1)$.

When the component at frequency k where $\hat{h}(k)$ is not close enough to $\hat{f}(k)$, i.e., $A(k) \neq 0$, $\exp\left(-|\pi k/2w_j|\right)$ would dominate $F_{lj}(\theta_j, k)$ for a small w_j. Intuitively, the gradient of low-frequency components dominates the training, thus, leading a fast convergence of low-frequency components. If w_j is larger, then, the dominance of the low frequency is less.

3.2 NTK Setting and Linear F-Principle

In general, it is difficult to analyze the convergence rate of each frequency due to its high dimensionality and nonlinearity. However, in a linear NTK regime [54], where the network has a width m approaching infinite and a scaling factor of $1/\sqrt{m}$, several works have explicitly shown the convergence rate of each frequency.

3.2.1 NTK Dynamics

One can consider the following gradient-descent flow dynamics of the empirical risk L_S of a network function $f(\cdot, \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ on a set of training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$

$$\begin{cases} \dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} L_S(\boldsymbol{\theta}), \\ \boldsymbol{\theta}(0) = \boldsymbol{\theta}_0, \end{cases} \tag{10}$$

where

$$L_S(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}(f(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i)^2. \tag{11}$$

Then, the training dynamics of the output function $f(\cdot, \boldsymbol{\theta})$ is

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} f(\boldsymbol{x}, \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}) \cdot \dot{\boldsymbol{\theta}} \\ &= -\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} L_S(\boldsymbol{\theta}) \\ &= -\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}) \cdot \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta})(f(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i) \\ &= -\sum_{i=1}^{n} K_m(\boldsymbol{x}, \boldsymbol{x}_i)(f(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i), \end{aligned}$$

where for time t the NTK evaluated at $(\boldsymbol{x}, \boldsymbol{x}') \in \Omega \times \Omega$ reads as

$$K_m(\boldsymbol{x}, \boldsymbol{x}')(t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}(t)) \cdot \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}', \boldsymbol{\theta}(t)). \tag{12}$$

At the NTK regime, we denote

$$K^*(\boldsymbol{x}, \boldsymbol{x}') := \lim_{m \to \infty} K_m(\boldsymbol{x}, \boldsymbol{x}')(t). \tag{13}$$

The gradient descent of the model thus becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(f(\boldsymbol{x},\boldsymbol{\theta}(t))-f(\boldsymbol{x})\Big)=-\sum_{i=1}^{n}K^*(\boldsymbol{x},\boldsymbol{x}_i)\Big(f(\boldsymbol{x}_i,\boldsymbol{\theta}(t))-f(\boldsymbol{x}_i)\Big). \tag{14}$$

Define the residual $\boldsymbol{u}(\boldsymbol{x},t)=f(\boldsymbol{x},\boldsymbol{\theta}(t))-f(\boldsymbol{x})$. Denote $\boldsymbol{X}:=(\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n)^{\mathrm{T}}\in\mathbb{R}^{n\times d}$ and $\boldsymbol{Y}:=(\boldsymbol{y}_1,\cdots,\boldsymbol{y}_n)^{\mathrm{T}}\in\mathbb{R}^n$ as the training data, $u(\boldsymbol{X},t):=(u(\boldsymbol{x}_1,t),\cdots,u(\boldsymbol{x}_n,t))\in\mathbb{R}^n$, $\nabla_{\boldsymbol{\theta}}f(\boldsymbol{X},\boldsymbol{\theta}(t)):=(\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_1,\boldsymbol{\theta}(t)),\cdots,\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_n,\boldsymbol{\theta}(t)))\in\mathbb{R}^{n\times M}$ (M is the number of parameters), and denote $K^*\in\mathbb{R}^{n\times n}$ as a matrix with

$$K^*_{i,j}=K^*(\boldsymbol{x}_i,\boldsymbol{x}_j). \tag{15}$$

Then, one can obtain

$$\frac{\mathrm{d}u(\boldsymbol{X},t)}{\mathrm{d}t}=-K^*u(\boldsymbol{X},t). \tag{16}$$

In a continuous form, one can define the empirical density $\rho(\boldsymbol{x})=\sum_{i=1}^{n}\delta(\boldsymbol{x}-\boldsymbol{x}_i)/n$ and further denote $u_\rho(\boldsymbol{x},t)=u(\boldsymbol{x},t)\rho(\boldsymbol{x})$. Therefore, the dynamics for u becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}u(\boldsymbol{x},t)=-\int_{\mathbb{R}^d}K^*(\boldsymbol{x},\boldsymbol{x}')u_\rho(\boldsymbol{x}',t)\mathrm{d}\boldsymbol{x}'. \tag{17}$$

This continuous form renders an integral equation analyzed in E et al. [33].

The convergence analysis of the dynamics in (16) can be done by performing eigen decomposition of K^*. The component in the sub-space of an eigen-vector converges faster with a larger eigen-value. A series of works further show that the eigen-vector with a larger eigen-value is lower frequency, therefore, providing a rigorous proof for the low-frequency bias of the DNN training process in the NTK regime for two-layer networks.

Consider a two-layer DNN

$$f(\boldsymbol{x},\boldsymbol{\theta})=\frac{1}{\sqrt{m}}\sum_{j=1}^{m}a_j\sigma(\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}+b_j), \tag{18}$$

where the vector of all parameters $\boldsymbol{\theta}$ is formed of the parameters for each neuron $(a_j,\boldsymbol{w}_j^{\mathrm{T}},b_j)^{\mathrm{T}}\in\mathbb{R}^{d+2}$ for $j\in[m]$. At the infinite neuron limit $m\to\infty$, the following linearization around initialization:

$$f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))=f(\boldsymbol{x};\boldsymbol{\theta}(0))+\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}(0))(\boldsymbol{\theta}(t)-\boldsymbol{\theta}(0)) \tag{19}$$

is an effective approximation of $f(\boldsymbol{x};\boldsymbol{\theta}(t))$, i.e., $f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))\approx f(\boldsymbol{x};\boldsymbol{\theta}(t))$ for any t, as demonstrated by both theoretical and empirical studies of NTKs [54, 65]. Note that, $f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))$, linear in $\boldsymbol{\theta}$ and nonlinear in $\boldsymbol{x}$, reserves the universal approximation power of $f(\boldsymbol{x};\boldsymbol{\theta}(t))$ at $m\to\infty$. In the following of this subsection, we do not distinguish $f(\boldsymbol{x};\boldsymbol{\theta}(t))$ from $f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))$.

3.2.2 Eigen Analysis for Two-Layer DNN with Dense Data Distribution

For a two-layer ReLU network, K^* enjoys good properties for the theoretical study. The exact form of K^* can be theoretically obtained [111]. Under the condition that training samples are distributed uniformly on a sphere, the expectation of the gram matrix for a two-layer ReLU network with respect to the samples can be explicitly computed, then, the spectrum of K^* can be obtained through spherical harmonic decomposition [111]. In a rough intuition, each

eigen-vector of K^* corresponds to a specific frequency. Based on such harmonic analysis, Cao et al. [18] and Basri et al. [7] estimate the eigen-values of K^*, i.e., the convergence rate of each frequency. For an arbitrary data distribution, it is often difficult to obtain the explicit form of the eigen decomposition of K^*. Basri et al. [6] further release the condition that data distributed uniformly on a sphere to that data distributed piecewise constant on a sphere but limit the result on the 1-D sphere. Similarly under the uniform distribution assumption in the NTK regime, Bordelon et al. [10] show that as the size of the training set grows, ReLU DNNs fit successively higher spectral modes of the target function. Empirical studies also validate that real data often align with the eigen-vectors that have large eigen-values, i.e., low-frequency eigen-vectors [5, 27, 63].

3.2.3 Linear F-Principle for Two-Layer Neural Network with Arbitrary Data Distribution

The condition of dense distribution, such as uniform on a sphere, is often non-realistic in training. Parallel work by Zhang et al. [125, 126] studies the evolution of each frequency for two-layer wide ReLU networks with any data distribution, including randomly discrete cases, and derives the linear F-Principle (LFP) model. Luo et al. [77] provide a rigorous version of Zhang et al. [125, 126] and extend the study of ReLU activation function in Zhang et al. [125, 126] to general activation functions. Luo et al. [77] circumvent the difficulty of eigen decomposition for arbitrary data distribution by analyzing the evolution of the network output in the Fourier space. The convergence rate of each frequency can be explicitly obtained, thus, its analysis can be extended to arbitrary data distribution.

The key idea is to perform the Fourier transform of the kernel $K^*(\boldsymbol{x}, \boldsymbol{x}')$ w.r.t. both $\boldsymbol{x}$ and $\boldsymbol{x}'$. To separate the evolution of each frequency, one has to assume the bias is significantly larger than 1. In numerical experiments, by taking the order of bias as the maximum of the input weight and the output weight, one can obtain an accurate approximation of two-layer DNNs with the LFP.

We use an LFP result for the two-layer ReLU network for intuitive understanding. Consider the residual $\boldsymbol{u}(\boldsymbol{x}, t) = f(\boldsymbol{x}, \boldsymbol{\theta}(t)) - f^*(\boldsymbol{x})$, one can obtain

$$\partial_t \hat{u} = -(\gamma(\boldsymbol{\xi}))^2 \hat{u_\rho}(\boldsymbol{\xi}) \tag{20}$$

with

$$(\gamma(\boldsymbol{\xi}))^2 = \frac{C_1}{\|\boldsymbol{\xi}\|^{d+3}} + \frac{C_2}{\|\boldsymbol{\xi}\|^{d+1}} \tag{21}$$

where $\hat{}$ is the Fourier transform, C_1 and C_2 are constants depending on the initial distribution of parameters, and $(\cdot)_\rho(\boldsymbol{x}) := (\cdot)(\boldsymbol{x})\rho(\boldsymbol{x})$. $\rho(\boldsymbol{x})$ is the data distribution, which can be a continuous function or $\rho(\boldsymbol{x}) = \Sigma_{i=1}^n \delta(\boldsymbol{x} - \boldsymbol{x}_i)/n$.

One can further prove that the long-time solution of (20) satisfies the following constrained minimization problem:

$$\begin{aligned} &\min_h \int \gamma(\boldsymbol{\xi})^{-2} \left|\hat{h}(\boldsymbol{\xi}) - \hat{h}_{\text{ini}}(\boldsymbol{\xi})\right|^2 \mathrm{d}\boldsymbol{\xi}, \\ &\text{s.t. } h(\boldsymbol{x}_i) = f^*(\boldsymbol{x}_i),\ i = 1, \cdots, n. \end{aligned} \tag{22}$$

Based on the equivalent optimization problem in (22), each decaying term for 1-D problems ($d = 1$) can be analyzed. When $1/\xi^2$ term dominates, the corresponding minimization

problem (22) which can be rewritten into the spatial domain yields

$$\begin{aligned} &\min_{h} \int \left|h^{'}(x) - h^{'}_{\text{ini}}(x)\right|^2 \mathrm{d}x, \\ &\text{s.t.} \quad h(\boldsymbol{x}_i) = f^*(\boldsymbol{x}_i), i = 1, \cdots, n, \end{aligned} \tag{23}$$

where $'$ indicates differentiation. For $h_{\text{ini}}(x) = 0$, (23) indicates a linear spline interpolation. Similarly, when $1/\xi^4$ dominates, $\int |h^{''}(x) - h^{''}_{\text{ini}}(x)|^2 \mathrm{d}x$ is minimized, indicating a cubic spline. In general, the above two power laws decay and coexist, giving rise to a specific mixture of linear and cubic splines. For high-dimensional problems, the model prediction is difficult to interpret because the order of differentiation depends on d and can be fractal. Similar analysis in the spatial domain can be found in the subsequent work in Jin and Montúfar [58].

Inspired by the variational formulation of the LFP model in (22), Luo et al. [76] propose a new continuum model for the supervised learning. This is a variational problem with a parameter $\alpha > 0$:

$$\min_{h \in \mathcal{H}} Q_\alpha[h] = \int_{\mathbb{R}^d} \langle \boldsymbol{\xi} \rangle^\alpha \left|\hat{h}(\boldsymbol{\xi})\right|^2 \mathrm{d}\boldsymbol{\xi}, \tag{24}$$

$$\text{s.t.} \quad h(\boldsymbol{x}_i) = y_i, \quad i = 1, \cdots, n, \tag{25}$$

where $\langle \boldsymbol{\xi} \rangle = (1 + \|\boldsymbol{\xi}\|^2)^{\frac{1}{2}}$ is the "Japanese bracket" of $\boldsymbol{\xi}$ and $\mathcal{H} = \{h(x) | \int_{\mathbb{R}^d} \langle \boldsymbol{\xi} \rangle^\alpha \left|\hat{h}(\boldsymbol{\xi})\right|^2 \mathrm{d}\boldsymbol{\xi} < \infty\}$.

Luo et al. [76] prove that $\alpha = d$ is a critical point. If $\alpha < d$, the variational problem leads to a trivial solution that is only non-zero at the training data point. If $\alpha > d$, the variational problem leads to a solution with certain regularity. The LFP model shows that a DNN is a convenient way to implement the variational formulation, which automatically satisfies the well-posedness condition.

Finally, we give some remarks on the difference between the eigen decomposition and the frequency decomposition. In the non-NTK regime, the eigen decomposition can be similarly analyzed but without informative explicit form. In addition, the study of bias from the perspective of eigen decomposition is very limited. For finite networks, which are practically used, the kernel evolves with training. Thus, it is hard to understand what kind of component converges faster or slower. The eigen mode of the kernel is also difficult to perceive. In contrast, the frequency decomposition is easy to interpret, and a natural approach is widely used in science.

3.3 Spectral Bias of Fully Connected Multi-Layer Networks

Luo et al. [78] consider fully connected multi-layer DNNs trained by a general loss function $\tilde{R}_{\mathcal{D}}(\boldsymbol{\theta})$ and measure the convergence of different frequencies by a mean squared loss. Similar to the filtering method, the approach of Luo et al. [78] is to decompose the frequency domain into a low-frequency part and a high-frequency part. The key idea is as follows. The Fourier spectrum of an activation function with a certain regularity would decay with a certain rate. This decay rate would lead to that the gradient of the loss function of a particular frequency would decay with frequency.

Based on the following assumptions, i.e., i) certain regularity of target function, sample distribution function, and activation function; ii) bounded training trajectory with loss convergence. Luo et al. [78] prove that the change of high-frequency loss over the total loss

decays with the separated frequency with a certain power, which is determined by the regularity assumption. A key ingredient in the proof is that the composition of functions still has a certain regularity, which renders the decay in the frequency domain. This result thus holds for general network structures with multiple layers. Aside from its generality, the characterization of the F-Principle is too coarse-grained to differentiate the effect network structure or the specialty of the DNNs, but only gives a qualitative understanding.

E et al. [33] present a continuous framework to study machine learning and suggest that the gradient flows of neural networks are nice flows, and they obey the F-Principle, basically because they are integral equations. The regularity of integral equations is higher, thus, leading to a faster decay in the Fourier domain.

4 Understanding and Studying DNN Based on F-Principle

In this section, we review how the F-Principle gains understandings of over-parameterized DNNs [122] and inspires the study of DNNs. First, we use a series of experiments to study what kind of settings can lead to the low-frequency bias. Second, we utilize the F-Principle to qualitatively and quantitatively study the generalization of DNNs. Third, we review the frequency approach for studying DNNs beyond the F-Principle. Finally, we review algorithms inspired by the F-Principle.

4.1 Empirical Study of Mechanisms Underlying Spectral Bias

4.1.1 Activation

The analysis in Sect. 3.1 shows the importance of the activation in the F-Principle. For most activations, such as tanh and ReLU, they monotonically decay in the frequency domain. Therefore, we can easily observe that the DNN training process follows the F-Principle, such as the example in Fig. 2. We can also design an activation that does not monotonically decay in the frequency domain but monotonically increases up to a high frequency, where we expect to observe the frequency convergence order may flip up to a certain frequency. We use the Ricker function with parameter a,

$$\frac{1}{15a}\pi^{1/4}\left(1-\left(\frac{x}{a}\right)^2\right)\exp\left(-\frac{1}{2}\left(\frac{x}{a}\right)^2\right). \tag{26}$$

With smaller a, the Ricker function decays from a higher frequency. We perform a similar learning task and use the same setting as Fig. 2. As shown in Fig. 6, in the first row, when the activation decays from a low frequency, we can clearly observe the low frequency converges faster; however, in the second row, when the activation decays from a high frequency, we cannot observe any frequency that converges faster, which is consistent with our analysis.

4.1.2 Frequency Weight in the Loss Function

The loss function form can affect the frequency convergence. For example, one can explicitly impose a large weight on some specific frequency component to accelerate the convergence of the frequency component. We consider two types of loss functions in learning the function as in Fig. 2, one is the common mean squared loss L_{nongrad}, and the other is one with an extra

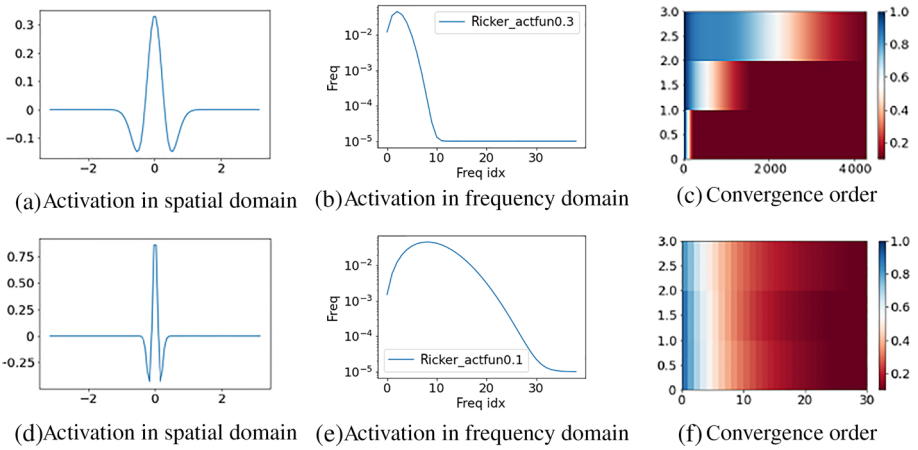


Fig. 6 Ricker activation function. $a = 0.3$ for the first row and $a = 0.1$ for the second row

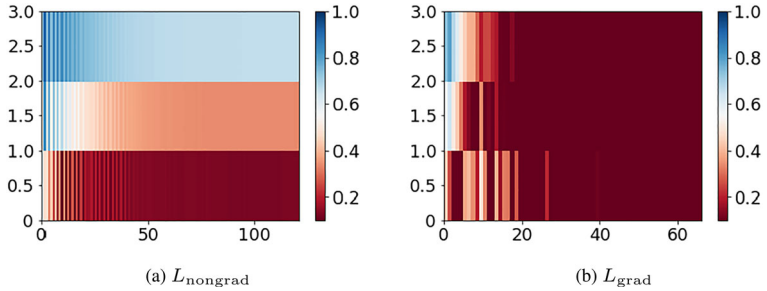


Fig. 7 The frequency convergence for networks with the loss functions L_{nongrad} and L_{grad}

loss of gradient L_{grad},

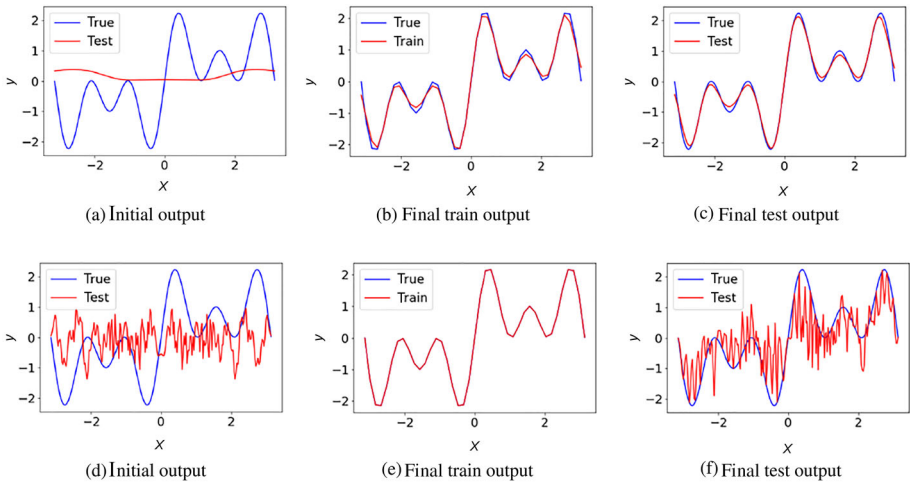
$$L_{\text{nongrad}} = \sum_{i=1}^{n} (f_\theta(x_i) - f^*(x_i))^2/n, \tag{27}$$

$$L_{\text{grad}} = L_{\text{nongrad}} + \sum_{i=1}^{n} (\nabla_x f_\theta(x_i) - \nabla_x f^*(x_i))^2/n. \tag{28}$$

As shown in Fig. 7, the high frequency converges much faster in the case of the loss with gradient information. The key reason is that, the Fourier transform of $\nabla_x f_\theta(x_i)$ is the product of the transform of $f_\theta(x_i)$ and frequency ξ, which is equivalent to adding more priority to the higher frequency.

4.1.3 The Joint Effect of Activation and Loss

The analysis in Sect. 3.1 shows that the frequency convergence behavior is the joint effect of activation and loss. A more detailed analysis is shown in (59). For common loss functions and activation functions, the DNN training process that follows the F-Principle can be easily observed. However, in some specific settings or tasks, such as solving PDEs where the loss function often contains gradient information, the F-Principle may not hold.



(a) Initial output (b) Final train output (c) Final test output

(d) Initial output (e) Final train output (f) Final test output

Fig. 8 DNN outputs with the Ricker activation function. $a = 0.3$ for the first row and $a = 0.1$ for the second row. Same experiments as Fig. 6

4.2 DNN that Violates F-Principle Produces Oscillated Output

To examine the utility of the F-Principle, we compare DNNs outputs for two experiments in Fig. 6. Note that the settings for the two examples in Fig. 6 are exactly the same except for the hyper-parameter a in the Ricker activation function. For smaller a, the output of the Ricker activation with small a is more oscillated than that of large a. Therefore, the initial output of the network with small a, i.e., the one that follows the F-Principle, is smooth (Fig. 8a), while the one with large a, i.e., the one that does not follow the F-Principle, is very oscillated (Fig. 8d). After training, the network that follows the F-Principle learns the training data well in (b), and the DNN output is smooth at test points in (c). However, for the one that does not follow the F-Principle, the DNN can also learn the training data well in (e), but it is very oscillated at test points in (f). It is also worth pointing out that the output of the network is always oscillated at test points during this training. Apparently, such oscillated output usually leads to bad generalization. This example shows the F-Principle is an important factor underlying the good generalization of DNNs.

4.3 Strength and Weakness

As demonstrated in the *Introduction* part, if the implicit bias or characteristic of an algorithm is consistent with the property of data, the algorithm generalizes well, otherwise not. By identifying the implicit bias of the DNNs in the F-Principle, we can have a better understanding of the strength and the weakness of deep learning, as demonstrated by Xu et al. [114] in the following.

DNNs often generalize well for real problems [122] but poorly for problems like fitting a parity function [87, 98] despite excellent training accuracy for all problems. The following demonstrates a qualitative difference between these two types of problems through *Fourier analysis* and uses the *F-Principle* to provide an explanation for different generalization performances of DNNs.

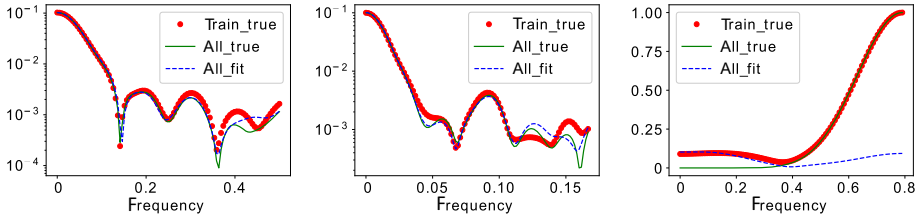


Fig. 9 Fourier analysis for different generalization abilities. The plot is the amplitude of the Fourier coefficient against frequency k. The red dots are for the training dataset, the green line is for the whole dataset, and the blue dashed line is for an output of well-trained DNN on the input of the whole dataset. For (**c**), $d = 10$. The training data are 200 randomly selected points. The settings of (**a**) and (**b**) are the same as the ones in Fig. 4. For (**c**), we use a tanh-DNN with widths 10-500-100-1, learning rate 0.000 5 under full batch-size training by Adam optimizer. The parameters of the DNN are initialized by a Gaussian distribution with mean 0 and standard deviation 0.05. Reprinted from Xu et al. [114]

Using the projection method in Sect. 2.2.1, one can obtain frequencies along the examined directions. For illustration, the Fourier transform of all MNIST/CIFAR10 data along the first principle component is shown in Figs. 9a, b for MNIST/CIFAR10, respectively. The Fourier transform of the training data (red dot) well overlaps with that of the total data (green) at the dominant low frequencies. As expected, the Fourier transform of the DNN output with bias of the low frequency, evaluated on both training and test data, also overlaps with the true Fourier transform at the low-frequency part. Due to the negligible high frequency of these problems, the DNNs generalize well.

However, DNNs generalize badly for high-frequency functions as follows. For the parity function $f(\boldsymbol{x}) = \prod_{j=1}^{d} x_j$ defined on $\Omega = \{-1, 1\}^d$, its Fourier transform is $\hat{f}(\boldsymbol{k}) = \frac{1}{2^d} \sum_{x\in\Omega} \prod_{j=1}^{d} x_j e^{-i2\pi \boldsymbol{k}\cdot\boldsymbol{x}} = (-i)^d \prod_{j=1}^{d} \sin 2\pi k_j$. Clearly, for $\boldsymbol{k} \in [-\frac{1}{4}, \frac{1}{4}]^d$, the power of the parity function concentrates at $\boldsymbol{k} \in \{-\frac{1}{4}, \frac{1}{4}\}^d$ and vanishes as $\boldsymbol{k} \to \boldsymbol{0}$, as illustrated in Fig. 9c for the direction of $\mathbf{1}_d$. Given a randomly sampled training dataset $S \subset \Omega$ with s points, the nonuniform Fourier transform on S is computed as $\hat{f}_S(\boldsymbol{k}) = \frac{1}{s} \sum_{x\in S} \prod_{j=1}^{d} x_j e^{-i2\pi \boldsymbol{k}\cdot\boldsymbol{x}}$. As shown in Fig. 9c, $\hat{f}(\boldsymbol{k})$ and $\hat{f}_S(\boldsymbol{k})$ significantly differ at low frequencies, caused by the well-known *aliasing* effect. Based on the F-Principle, as demonstrated in Fig. 9c, these artificial low-frequency components will be first captured to explain the training samples, whereas the high-frequency components will be compromised by DNN, leading to a bad generalization performance as observed in experiments.

The F-Principle implicates that among all the functions that can fit the training data, a DNN is implicitly biased during the training towards a function with more power at low frequencies, which is consistent with the implication of the equivalent optimization problem (22). The distribution of power in the Fourier domain of above two types of problems exhibits significant differences, which results in different generalization performances of DNNs according to the F-Principle.

Ma et al. [80] show that the F-Principle may be a general mechanism behind the slow deterioration phenomenon in the training of DNNs, where the effect of the "double descent" is washed out. Sharma and Ross [99] utilize the low-frequency bias of DNNs to study effectiveness of an iris recognition DNN. Chen et al. [20] show that under the same computational budget, a MuffNet is a better universal approximator for functions containing high-frequency components, thus, better for mobile deep learning. Zhu et al. [130] utilize the F-Principle

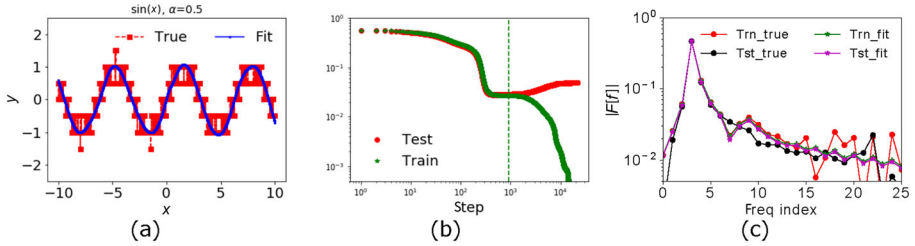


Fig. 10 Effect of early-stopping on contaminated data. The training set and the test set consist of 300 and 6 000 data points evenly sampled in $[-10, 10]$, respectively. **a** The sampled values of the test set (red square dashed line) and DNN outputs (blue solid line) at the turning step. **b** Loss functions for the training set (green stars) and test set (red dots) at different recording steps. The green dashed line is drawn at the turning step, where the best generalization performance is achieved. **c** The Fourier transform of the true data for the training set (red) and test set (black), and the Fourier transform of the DNN output for the training set (green), and test set (magenta) at the turning step. Reprinted from Xu et al. [115]

to help understand why high frequency is a limit when DNNs are used to solve the spectral deconvolution problem. Chakrabarty [19] utilizes the idea of the F-Principle to study the spectral bias of the deep image prior.

4.4 Early Stopping

When the training data are contaminated by noise, the early-stopping method is usually applied to avoid overfitting in practice [72]. By the F-Principle, early-stopping can help avoid fitting the noisy high-frequency components. Thus, it naturally leads to a well-generalized solution. Xu et al. [115] use the following example for illustration.

As shown in Fig. 10a, the data are sampled from a function with noise. The DNN can well fit the sampled training set as the loss function of the training set decreases to a very small value (green stars in Fig. 10b). However, the loss function of the test set first decreases and then increases (red dots in Fig. 10b). In Fig. 10c, the Fourier transform for the training data (red) and the test data (black) only overlaps around the dominant low-frequency components. Clearly, the high-frequency components of the training set are severely contaminated by noise. Around the turning step—where the best generalization performance is achieved, indicated by the green dashed line in Fig. 10b—the DNN output is a smooth function (blue line in Fig. 10a) in the spatial domain and well captures the dominant peak in the frequency domain (Fig. 10c). After that, the loss function of the test set increases as DNN start to capture the higher frequency noise (red dots in Fig. 10b). These phenomena conform with our analysis that early-stopping can lead to a better generalization performance of DNNs as it helps avoid fitting the noisy high-frequency components of the training set.

As a low-frequency function is more robust w.r.t. input than a high-frequency function, the early-stopping can also enhance the robustness of the DNN. This effect is consistent with the study by Li et al. [66], which shows that a two-layer DNN, trained only on the input weight and early stopped, can reconstruct the true labels from noisy data.

4.5 Quantitative Understanding in NTK Regime

The static minimization problem (22) defines an FP-energy $E_\gamma(h) = \int \gamma^{-2}|\hat{h}|^2 \mathrm{d}\boldsymbol{\xi}$ that quantifies the preference of the LFP model among all its steady states. Since $\gamma(\boldsymbol{\xi})^{-2}$ is an

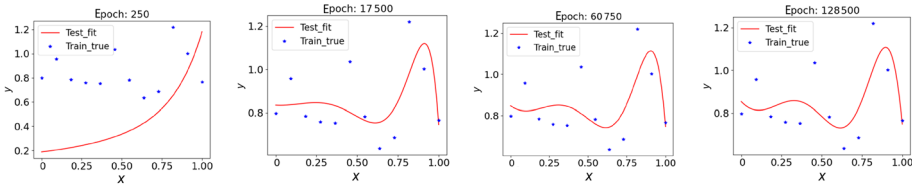


Fig. 11 Using mean squared loss and gradient descent to find the solution of a polynomial interpolation with order 11 and 12 equispaced points

increasing function, say $\gamma(\boldsymbol{\xi})^{-2} = \|\boldsymbol{\xi}\|^{d+1}$, the FP-energy $\int \|\boldsymbol{\xi}\|^{d+1}|\hat{h}|^2 \mathrm{d}\boldsymbol{\xi}$ amplifies the high frequencies while diminishing low frequencies. By minimizing $E_\gamma(h)$, problem (22) gives rise to a low frequency fitting, instead of an arbitrary one, of training data. By intuition, if target f^* is indeed the low frequency dominant, then h_∞ likely well approximates f^* at unobserved positions.

To theoretically demonstrate above intuition, Luo et al. [77] derive the following, an estimate of the generalization error of h_∞ using the a priori error estimate technique [32]. Since $h(\boldsymbol{x}) = f^*(\boldsymbol{x})$ is a viable steady state, $E_\gamma(h_\infty) \leqslant E_\gamma(f^*)$ by the minimization problem. Using this constraint on h_∞, one can obtain that, with probability of at least $1-\delta$,

$$\mathbb{E}_{\boldsymbol{x}}(h_\infty(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 \leqslant \frac{E_\gamma(f^*)}{\sqrt{n}} C_\gamma \left(2 + 4\sqrt{2\log(4/\delta)}\right), \tag{29}$$

where C_γ is a constant depending on γ. Error reduces with more training data as expected with a decay rate $1/\sqrt{n}$ similar to the Monte-Carlo method. Importantly, because $E_\gamma(f^*)$ strongly amplifies high frequencies of f^*, the more high-frequency components the target function f^* possesses, the worse h_∞ may generalize.

Note that the error estimate is also consistent with another result [2] published at similar time. Arora et al. [2] prove that the generalization error of the two-layer ReLU network in the NTK regime found by GD is at most

$$\sqrt{\frac{2Y^{\mathrm{T}}(K^*)^{-1}Y}{n}}, \tag{30}$$

where K^* is defined in (15), $Y \in \mathbb{R}^n$ is the labels of n training data. If the data Y is dominated more by the component of the eigen-vector that has a small eigen-value, then, the above quantity is larger. Since in the NTK regime the eigen-vector that has a small eigen-value corresponds to a higher frequency, the error bound in (29) is larger, consistent with (30).

4.6 Frequency Perspective for Understanding Experimental Phenomena

Compression phase Xu et al. [115] explain the compression phase in the information plane, proposed by Shwartz-Ziv and Tishby [101], by the F-Principle as follows. The entropy or information quantifies the possibility of output values, i.e., more possible output values lead to a higher entropy. In learning a discretized function, the DNN first fits the continuous low-frequency components of the discretized function, i.e., large entropy state. Then, the DNN output tends to be discretized as the network gradually captures the high-frequency components, i.e., entropy decreasing. Thus, the compression phase appears in the information plane.

Increasing complexity The F-Principle also explains the increasing complexity of the DNN output during the training. For common small initialization, the initial output of a DNN is often close to zero. The F-Principle indicates that the output of DNN contains higher and higher frequencies during the training. As frequency is a natural concept to characterize the complexity of a function, the F-Principle indicates that the complexity of the DNN output increases during the training. This increasing complexity of the DNN output during training is consistent with previous studies and subsequent works [3, 42, 49, 59, 60, 86].

Deep frequency principle Xu and Zhou [116] propose a deep F-Principle to understand an effect of depth in accelerating the training. For a DNN, the effective target function of the l-th hidden layer can be understood in the following way. Its input is the output of the $(l-1)$-th layer. The part from the l-th layer is to learn the mapping from the output of the $(l-1)$-th layer to the true labels. Therefore, the effective target function of the l-th hidden layer consists of the output of the $(l-1)$-th layer and the true labels. Xu and Zhou [116] empirically find a deep F-Principle: the effective target function for deeper hidden layer biases towards lower frequency during the training. Due to the F-Principle, this empirical study provides a rationale for understanding why depth can accelerate the training.

Frequency approach Camuto et al. [17] show that the effect of Gaussian noise injections to each hidden layer output is equivalent to a penalty of high-frequency components in the Fourier domain. Rabinowitz [91], and Deng and Zhang [25] use the F-Principle as one of typical phenomena to study the difference between the normal learning and the meta-learning. Chen et al. [21] show the F-Principle holds in a broad learning system. Schwarz et al. [97] study the frequency bias of generative models.

4.7 Inspiring the Design of Algorithm

In addition to scientific computing reviewed in Sect. 5.4, to accelerate the convergence of high-frequency, different approaches are developed in various applications. Some examples are listed in the following.

Agarwal et al. [1] and Liang et al. [71] also design different types of activation functions. Campo et al. [16] use a frequency filter to help reduce the interdependency between the low frequency and the (harder to learn) high-frequency components of the state-action value approximation to achieve better results in reinforcement learning. Several works use multi-scale input by projecting data into a high-dimensional space with a set of sinusoids in efficiently representing complex 3D objects and scenes [8, 43, 46, 50, 84, 88, 90, 106, 109, 112, 129]. Tancik et al. [105] use meta-learning to obtain a good initialization for fast and effective image restoration. Several works utilize frequency-aware information to improve the quality of high-frequency details of images generated by neural networks [22, 56, 69, 118]. Xi et al. [110] argue that the performance improvement in low-resolution image classification is affected by the inconsistency of learning between low-frequency components and high-frequency components, and Xi et al. [110] propose a network structure to overcome this inconsistent issue.

The F-Principle shows that DNNs quickly learn the low-frequency part, which is often dominated in the real data and more robust. At the early stage, the DNN is similar to a linear model [51, 60]. Some works take advantage of DNN at early training stage to save training computation cost. The original lottery ticket network [38] requires a full training of DNN, which has a very high computational cost. Most computation is used to capture the high frequency while the high frequency may be not important in many cases. You et al. [119] show that a small but critical subnetwork emerges at the early training stage (Early-Bird ticket),

and the performance of training this small subnetwork with the same initialization is similar to the training of the full network, thus, saving significant energy for training networks. Fu et al. [39, 40] utilize the robustness of the low frequency by applying the low-precision for early training stage to save computational cost without sacrificing the generalization performance.

5 F-Principle for DNN-Based Methods for Solving PDEs

Recently, DNN-based approaches have been actively explored for a variety of scientific computing problems, e.g., solving high-dimensional partial differential equations [29, 31, 37, 44, 47, 48, 61, 104] and molecular dynamics (MD) simulations [45]. For solving PDEs, one can use DNNs to parameterize the solution of a specific PDE [26, 30, 93, 120] or the operator of a type of PDE [37, 70, 74, 123]. An overview of using DNN to solve high-dimensional PDEs can be found in E et al. [29]. We would focus on the former approach.

The F-Principle is an important feature of DNN-based algorithms. In this section, we will first review the frequency convergence difference between the DNN-based algorithm and conventional methods, i.e., iterative methods and finite-element methods. Then, we rationalize their difference and understand the F-Principle from the iterative perspective. Finally, we review algorithms developed for overcoming the challenge of the high frequency in DNN-based methods.

5.1 Parameterize the Solution of a PDE

For intuitive illustration, we use Poisson's equation as an example, which has broad applications in mechanical engineering and theoretical physics [36],

$$-\Delta u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \tag{31}$$

with the boundary condition

$$u(\boldsymbol{x}) = \tilde{g}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega. \tag{32}$$

One can use a neural network $u(\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the set of DNN parameters. A deep Ritz approach [34] utilizes the following variational problem:

$$u^* = \arg\min_v J(v), \tag{33}$$

where the solution of the above minimization problem can be proved to be the solution of the Poisson's problem and the energy functional is defined as

$$J(v) = \int_\Omega \left(\frac{1}{2}|\nabla v|^2 + V(\boldsymbol{r})v^2\right) \mathrm{d}\boldsymbol{r} - \int_\Omega g(\mathbf{r})v(\mathbf{r})\mathrm{d}\boldsymbol{r} \triangleq \int_\Omega E(v(\mathbf{r}))\mathrm{d}r. \tag{34}$$

In numerical computing, the solution of the Poisson's problem is parameterized by a DNN $u(\boldsymbol{x};\boldsymbol{\theta})$, the target functional is discretized in the form of the first part of (35). The Dirichlet boundary condition is treated as an L^2 penalty and discretized as the second part of (35). That is,

$$L_{\mathrm{ritz}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{\boldsymbol{x}\in S}(|\nabla u(\boldsymbol{x};\boldsymbol{\theta})|^2/2 - g(\boldsymbol{x})u(\boldsymbol{x};\boldsymbol{\theta})) + \frac{\beta}{\tilde{n}}\sum_{\boldsymbol{x}\in\tilde{S}}(u(\boldsymbol{x};\boldsymbol{\theta}) - \tilde{g}(\boldsymbol{x}))^2, \tag{35}$$

where S is the sample set from Ω, n is the sample size, and $\tilde{n}$ indicates the sample set from $\partial\Omega$. The second penalty term with a weight β is to enforce the boundary condition.

A more direct method, also known as the physics-informed neural network (PINN) or least squared method, use the following loss function. In an alternative approach, one can simply uses the loss function of the Least Squared Error (LSE)

$$L_{\mathrm{LSE}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{\boldsymbol{x}\in S}(\Delta u(\boldsymbol{x};\boldsymbol{\theta}) + g(\boldsymbol{x}))^2 + \frac{\beta}{\tilde{n}}\sum_{\boldsymbol{x}\in \tilde{S}}(u(\boldsymbol{x};\boldsymbol{\theta}) - \tilde{g}(\boldsymbol{x}))^2. \tag{36}$$

To see the learning accuracy, one can compute the distance between $u(\boldsymbol{x};\boldsymbol{\theta})$ and u_{true},

$$\mathrm{MSE}(u(\boldsymbol{x};\boldsymbol{\theta}), u_{\mathrm{true}}(\boldsymbol{x})) = \frac{1}{n+\tilde{n}}\sum_{\boldsymbol{x}\in S\cup\tilde{S}}(u(\boldsymbol{x};\boldsymbol{\theta}) - u_{\mathrm{true}}(\boldsymbol{x}))^2. \tag{37}$$

5.2 Difference from Conventional Algorithms

5.2.1 Iterative Methods

A stark difference between a DNN-based solver and the Jacobi method during the training/iteration is that DNNs learn the solution from low- to high-frequencies [114], while the Jacobi method learns the solution from high to low frequencies. Therefore, DNNs would suffer from the high-frequency curse.

Jacobi method Before we show the difference between a DNN-based solver and the Jacobi method, we illustrate the procedure of the Jacobi method.

Consider a 1-D Poisson's equation:

$$-\Delta u(x) = g(x), \quad x \in \Omega \triangleq (-1, 1), \tag{38}$$

$$u(-1) = u(1) = 0. \tag{39}$$

$[-1, 1]$ is uniformly discretized into $n+1$ points with the grid size $h = 2/n$. The Poisson's equation in (38) can be solved by the central difference scheme,

$$-\Delta u_i = -\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = g(x_i), \quad i = 1, 2, \cdots, n, \tag{40}$$

resulting a linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{g}, \tag{41}$$

where

$$\boldsymbol{A} = \begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & & & & \vdots \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}_{(n-1)\times(n-1)}, \tag{42}$$

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix}, \quad \boldsymbol{g} = h^2 \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{n-2} \\ g_{n-1} \end{pmatrix}, \quad x_i = 2\frac{i}{n}. \tag{43}$$

A class of methods to solve this linear system is iterative schemes, for example, the Jacobi method. Let $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$, where $\boldsymbol{D}$ is the diagonal of $\boldsymbol{A}$, and $\boldsymbol{L}$ and $\boldsymbol{U}$ are the strictly lower and upper triangular parts of $-\boldsymbol{A}$, respectively. Then, we obtain

$$\boldsymbol{u} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})\boldsymbol{u} + \boldsymbol{D}^{-1}\boldsymbol{g}. \tag{44}$$

At step $t \in \mathbb{N}$, the Jacobi iteration reads as

$$\boldsymbol{u}^{t+1} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})\boldsymbol{u}^{t} + \boldsymbol{D}^{-1}\boldsymbol{g}. \tag{45}$$

We perform the standard error analysis of the above iteration process. Denote $\boldsymbol{u}^*$ as the true value obtained by directly performing inverse of $\boldsymbol{A}$ in (41). The error at step $t+1$ is $\boldsymbol{e}^{t+1} = \boldsymbol{u}^{t+1} - \boldsymbol{u}^*$. Then, $\boldsymbol{e}^{t+1} = \boldsymbol{R}_J \boldsymbol{e}^t$, where $\boldsymbol{R}_J = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})$. The converging speed of $\boldsymbol{e}^t$ is determined by the eigen-values of $\boldsymbol{R}_J$, that is,

$$\lambda_k = \lambda_k(\boldsymbol{R}_J) = \cos\frac{k\,\pi}{n}, \quad k = 1, 2, \cdots, n-1, \tag{46}$$

and the corresponding eigen-vector $\boldsymbol{v}_k$'s entry is

$$v_{k,i} = \sin\frac{ik\,\pi}{n}, i = 1, 2, \cdots, n-1. \tag{47}$$

Therefore, we can write

$$\boldsymbol{e}^t = \sum_{k=1}^{n-1} \alpha_k^t \boldsymbol{v}_k, \tag{48}$$

where α_k^t can be understood as the magnitude of $\boldsymbol{e}^t$ in the direction of $\boldsymbol{v}_k$. Then,

$$\boldsymbol{e}^{t+1} = \sum_{k=1}^{n-1} \alpha_k^t \boldsymbol{R}_J \boldsymbol{v}_k = \sum_{k=1}^{n-1} \alpha_k^t \lambda_k \boldsymbol{v}_k, \tag{49}$$

$$\alpha_k^{t+1} = \lambda_k \alpha_k^t.$$

Therefore, the converging rate of $\boldsymbol{e}^t$ in the direction of $\boldsymbol{v}_k$ is controlled by λ_k. Since

$$\cos\frac{k\,\pi}{n} = -\cos\frac{(n-k)\,\pi}{n}, \tag{50}$$

the frequencies k and $(n-k)$ are closely related and converge with the same rate. Consider the frequency $k < n/2$, λ_k is larger for the lower frequency. Therefore, the lower frequency converges more slowly in the Jacobi method.

Numerical experiments Xu et al. [114] consider the example with $g(x) = \sin(x) + 4\sin(4x) - 8\sin(8x) + 16\sin(24x)$ such that the exact solution $u_{\text{ref}}(x)$ has several high frequencies. After training with the Ritz loss, the DNN output well matches the analytical solution u_{ref}. For each frequency k, we define the relative error as

$$\Delta_F(\boldsymbol{k}) = |\hat{u}_{\boldsymbol{\theta}}(\boldsymbol{k}) - \hat{u}_{\text{true}}(\boldsymbol{k})|/\hat{u}_{\text{true}}(\boldsymbol{k}).$$

Focusing on the convergence of three peaks (inset of Fig. 12a) in the Fourier transform of u_{ref}, as shown in Fig. 12b, low frequencies converge faster than high frequencies as predicted by the F-Principle. For comparison, Xu et al. [114] also use the Jacobi method to solve problem (38). High frequencies converge faster in the Jacobi method, as shown in Fig. 12c.

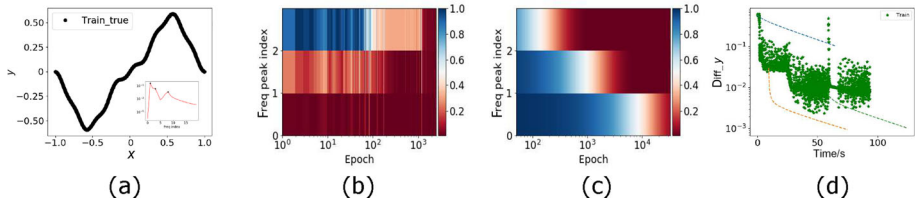


Fig. 12 Poisson's equation. **a** $u_{\text{ref}}(x)$. Inset: $|\hat{u}_{\text{ref}}(k)|$ as a function of frequency. Frequencies peaks are marked with black dots. **b**, **c** $\Delta_F(k)$ computed on the inputs of training data at different epochs for the selected frequencies for DNN (**b**) and Jacobi (**c**). **d** $\|h - u_{\text{ref}}\|_\infty$ at different running time. Green stars indicate $\|h - u_{\text{ref}}\|_\infty$ using DNN alone. The dashed lines indicate $\|h - u_{\text{ref}}\|_\infty$ for the Jacobi method with different colors indicating initialization by different timing of the DNN training. Xu et al. [114] use a DNN with widths 1-4000-500-400-1 and full batch training by Adam optimizer [62]. The learning rate is 0.000 5. β is 10. The parameters of the DNN are initialized following a Gaussian distribution with mean 0 and standard deviation 0.02. Reprinted from Xu et al. [114]

As a demonstration, Xu et al. [114] further propose that DNN can be combined with conventional numerical schemes to accelerate the convergence of low frequencies for computational problems. First, Xu et al. [114] solve the Poisson's equation in (38) by DNN with M optimization steps (or epochs). Then, Xu et al. [114] use the Jacobi method with the new initial data for the further iterations. A proper choice of M is indicated by the initial point of orange dashed line, in which low frequencies are quickly captured by the DNN, followed by fast convergence in high frequencies of the Jacobi method. A similar idea of using DNN as the initial guess for conventional methods is proved to be effective in later works [52].

This example illustrates a cautionary tale that, although DNNs have a clear advantage, using DNNs alone may not be the best option because of its limitation of the slow convergence at high frequencies. Taking advantage of both DNNs and conventional methods to design faster schemes could be a promising direction in scientific computing problems.

5.2.2 Ritz-Galerkin (R-G) Method

Wang et al. [108] study the difference between R-G method and DNN methods, reviewed as follows.

R-G method We briefly introduce the R-G method [12]. For problem (38), we construct a functional

$$J(u) = \frac{1}{2}a(u,u) - (g,u), \tag{51}$$

where

$$a(u,v) = \int_\Omega \nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \quad (g,v) = \int_\Omega g(\boldsymbol{x})v(\boldsymbol{x})\mathrm{d}\boldsymbol{x}.$$

The variational form of problem (38) is the following:

$$\text{find } u \in H_0^1(\Omega), \text{ s.t. } J(u) = \min_{v \in H_0^1(\Omega)} J(v). \tag{52}$$

The weak form of (52) is to find $u \in H_0^1(\Omega)$ such that

$$a(u,v) = (g,v), \quad \forall\, v \in H_0^1(\Omega). \tag{53}$$

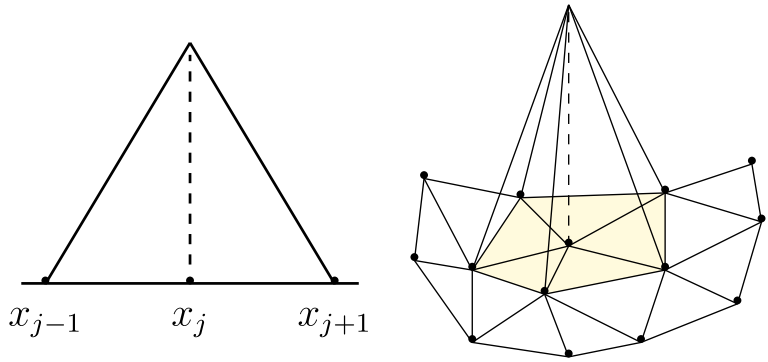


Fig. 13 The finite-element basis function in one dimension and two dimensions. Reprinted from Wang et al. [108]

The problem (38) is the strong form if the solution $u \in H_0^2(\Omega)$. To numerically solve (53), we now introduce the finite-dimensional space U_h to approximate the infinite-dimensional space $H_0^1(\Omega)$. Let $U_h \subset H_0^1(\Omega)$ be a sub-space with a sequence of basis functions $\{\phi_1, \phi_2, \cdots, \phi_m\}$. The numerical solution $u_h \in U_h$ that we will find can be represented as

$$u_h = \sum_{k=1}^{m} c_k \phi_k, \tag{54}$$

where the coefficients $\{c_i\}$ are the unknown values that we need to solve. Replacing $H_0^1(\Omega)$ by U_h, both problems (52) and (53) can be transformed to solve the following system:

$$\sum_{k=1}^{m} c_k a(\phi_k, \phi_j) = (g, \phi_j), \quad j = 1, 2, \cdots, m. \tag{55}$$

From (55), we can calculate c_i, and then obtain the numerical solution u_h. We usually call (55) the R-G equation.

For different types of basis functions, the R-G method can be divided into the finite-element method (FEM) and spectral method (SM) and so on. If the basis functions $\{\phi_i(\boldsymbol{x})\}$ are local, namely, they are compactly supported, this method is usually taken as the FEM. Assume that Ω is a polygon, and we divide it into finite-element grid $\mathcal{T}_h$ by simplex, $h = \max_{\tau \in \mathcal{T}_h} \text{diam}(\tau)$. A typical finite-element basis is the linear hat basis function, satisfying

$$\phi_k(\boldsymbol{x_j}) = \delta_{kj}, \quad \boldsymbol{x_j} \in \mathcal{N}_h, \tag{56}$$

where $\mathcal{N}_h$ stands for the set of the nodes of grid $\mathcal{T}_h$. The schematic diagram of the basis functions in one dimension and two dimensions is shown in Fig. 13. On the other hand, if we choose the global basis function such as the Fourier basis or Legendre basis [100], we call the R-G method spectral method.

The error estimate theory of the R-G method has been well established. Under suitable assumption on the regularity of the solution, the linear finite-element solution u_h has the following error estimate:

$$\|u - u_h\|_1 \leqslant C_1 h \|u\|_2,$$

where the constant C_1 is independent of the grid size h. The spectral method has the following error estimate:

$$\|u - u_h\| \leqslant \frac{C_2}{m^s},$$

where C_2 is a constant and the exponent s depends only on the regularity (smoothness) of the solution u. If u is smooth enough and satisfies certain boundary conditions, the spectral method has the spectral accuracy.

Different learning results DNNs with the ReLU activation function can be proved to be equivalent with a finite element method in the sense of approximation [47]. However, the learning results have a stark difference. To investigate the difference, we utilize a control experiment, that is, solving PDEs given n sample points and controlling the number of bases in the R-G method and the number of neurons in DNN equal m. Although not realistic in the common usage of the R-G method, we choose the case $m > n$ because the two methods are completely different in such a situation especially when $m \to \infty$. Then, replacing the integral on the r.h.s. of (55) with the form of the MC integral formula, we obtain

$$\sum_{k=1}^{m} c_k a(\phi_k, \phi_j) = \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{x}_i)\phi_j(\boldsymbol{x}_i), \quad j = 1, 2, \cdots, m. \tag{57}$$

We consider the 2-D case

$$\begin{cases} -\Delta u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in (0,1)^2, \\ u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial(0,1)^2, \end{cases}$$

where $\boldsymbol{x} = (x, y)$ and we know the values of g at $n = 5^2$ points sampled from the function $g(\boldsymbol{x}) := g(x, y) = 2\,\pi^2 \sin(\pi\, x) \sin(\pi\, y)$. For a large m, Figs. 14a, b plot the R-G solutions with the Legendre basis and piecewise linear basis function. It can be seen that the numerical solution is a function with the strong singularity. However, Figs. 14c, d show that the two-layer DNN solutions are stable without singularity for large m.

The smooth solution of DNN, especially when the neuron number is large, can be understood through the low-frequency bias, such as the analysis shown in the LFP theory. This helps understand the wide application of DNN in solving PDEs. For example, the low-frequency bias intuitively explains why DNN solves a shock wave by a smooth solution in Michoski et al. [83].

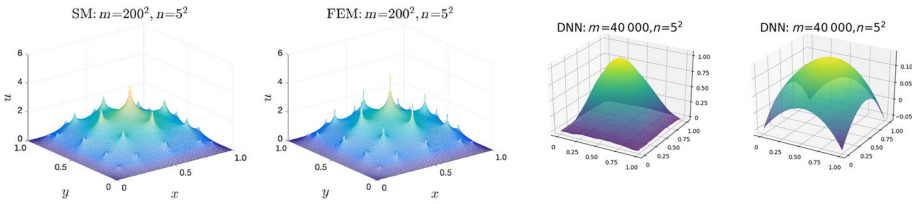
For the R-G method, the following theorem explains why there is singularity in the 2-D case when m is large.

Theorem 1 *When $m \to \infty$, the numerical method (57) is solving the problem*

$$\begin{cases} -\Delta u(\boldsymbol{x}) = \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} \delta(\boldsymbol{x} - \boldsymbol{x}_i) g(\boldsymbol{x}_i), & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \end{cases} \tag{58}$$

where $\delta(x)$ represents the Dirac delta function.

This theorem shows that if we consider an over-parameterized FEM, it solves the Green's function of the PDE. In Poisson's problem, the 2-D Green's function has a singularity, thus, leading to a singular solution. However, for DNN, due to the F-Principle of the low-frequency preference, the solution is always relatively smooth. Although there exists equivalence between the DNN-based algorithm and conventional methods for solving PDEs in the sense of approximation, it is important to take the implicit bias when analyzing the learning results of DNNs.



(a) SM (b) FEM (c) ReLU-DNN (d) Sin-DNN

Fig. 14 R-G solutions (**a**, **b**) and two-layer DNN solutions (**c**, **d**). Reprinted from Wang et al. [108]

5.3 Understanding F-Principle by Comparing the Differential Operator and the Integrator Operator

In this part, inspired by the analysis in E et al. [33], we use a very non-rigorous derivation to intuitively understand why DNN follows the F-Principle while the Jacobi method does not and shows a connection between these two methods.

Consider the elliptic equation in the one dimension:

$$\begin{cases} Lu := -\Delta u = g, \quad g \in L^2([0,1]), \\ u(0) = 0, \\ u(1) = 0. \end{cases}$$

Suppose that g is sufficiently smooth and thus u is smooth, $N >> 1, h = 1/N, x_i = ih = \frac{i}{N}, i = 0, \cdots, N$.

Let

$$\boldsymbol{v} = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_N \end{pmatrix} = \begin{pmatrix} u(0) \\ u(h) \\ \vdots \\ u(1) \end{pmatrix},$$

$v_0 = v_N = 0$. Then we have

$$(A_N \boldsymbol{v})_i = \frac{1}{h^2}\left(-v_{i-1} + 2v_i - v_{i+1}\right) = g_i, i = 1, \cdots, N-1,$$

$$A_N \boldsymbol{v} = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-1} \end{pmatrix},$$

$k = 1, \cdots, N-1$ (mode), $i = 1, \cdots, N-1$, which can be extended to $i = 0, N$ by considering the boundary conditions.

Define $(\boldsymbol{w}_k)_i := \sin\frac{ik\,\pi}{N}$, $(\boldsymbol{w}_k)_0 := (\boldsymbol{w}_k)_N := 0$,

$$\begin{aligned}(A_N \boldsymbol{w}_k)_i &= \frac{1}{h^2}\left(-(\boldsymbol{w}_k)_{i-1} + 2(\boldsymbol{w}_k)_i - (\boldsymbol{w}_k)_{i+1}\right) \\ &= \frac{1}{h^2}\left(-\sin\frac{(i-1)k\,\pi}{N} + 2\sin\frac{ik\,\pi}{N} - \sin\frac{(i+1)k\,\pi}{N}\right)\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{h^2}\left(-2\sin\frac{ik\,\pi}{N}\cos\frac{k\,\pi}{N} + 2\sin\frac{ik\,\pi}{N}\right)\\
&= (\boldsymbol{w}_k)_i\,\frac{2}{h^2}\left(1-\cos\frac{k\,\pi}{N}\right)\\
&= \frac{4}{h^2}\sin^2\frac{k\,\pi}{2N}\,(\boldsymbol{\omega}_k)_i \quad i = 1,\cdots,N-1.
\end{aligned}
$$

Thus, we have

$$
A_N\boldsymbol{w}_k = \lambda_k\boldsymbol{w}_k,
$$

where $\lambda_k = \frac{4}{h^2}\sin^2\frac{k\,\pi}{N}$, $k = 1,\cdots,N-1$.

As the analysis in the Jacobi iteration in Sect. 5.2.1, for the differential operator A_N, the lower frequency converges more slowly. This can also be revealed by optimizing $\boldsymbol{v}$ through a mean square error as follows. Define $\boldsymbol{e} = \boldsymbol{v} - \boldsymbol{u}^*$ as the error on the discretized grid points, and

$$
R_A := \frac{1}{2N}\left\|A_N\boldsymbol{v} - g\right\|^2.
$$

By the gradient-descent flow, we have

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{v} &= -\nabla_{\boldsymbol{v}}R_A\\
&= -\frac{1}{N}A_N\left(A_N\boldsymbol{v} - \boldsymbol{g}\right),
\end{aligned}
$$

then,

$$
\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{e} = -\frac{1}{N}A_N^2\boldsymbol{e}.
$$

Similarly, we obtain that the low frequency converges more slowly.

Then, we consider $u(\boldsymbol{x};\boldsymbol{\theta})$ be the NN function parameterized by $\boldsymbol{\theta}$ to approximate the PDE solution. The loss function is similarly defined:

$$
\begin{aligned}
R_N :&= \frac{1}{2N}\sum_{i=1}^{N-1}\left(\left(A_N u\right)(x_i,\boldsymbol{\theta}) - g\left(x_i\right)\right)^2 + \frac{1}{2}u\left(x_0,\boldsymbol{\theta}\right)^2 + \frac{1}{2}u\left(x_N,\boldsymbol{\theta}\right)^2\\
&= \frac{1}{2N}\left\|A_N u - g\right\|^2 + \frac{1}{2}u\left(x_0,\boldsymbol{\theta}\right)^2 + \frac{1}{2}u\left(x_N,\boldsymbol{\theta}\right)^2.
\end{aligned}
$$

The gradient flow w.r.t. $\boldsymbol{\theta}$ is

$$
\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}}R_N(\boldsymbol{\theta}).
$$

Then, the evolution of $u(\boldsymbol{x}, \boldsymbol{\theta})$ is

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}u\left(x_i, \boldsymbol{\theta}\right) =& -\nabla_{\boldsymbol{\theta}} u\left(x_i, \boldsymbol{\theta}\right) \cdot \nabla_{\boldsymbol{\theta}} R_N(\boldsymbol{\theta}) \\ =& -\nabla_{\boldsymbol{\theta}} u\left(x_i, \boldsymbol{\theta}\right) \left[\frac{1}{N}\sum_{j=1}^{N-1}\left(\left(A_N^2 u\right)_j - \left(A_N g\right)_j\right)\nabla_{\boldsymbol{\theta}} u\left(x_j, \boldsymbol{\theta}\right)\right. \\ &\left. + u\left(x_0, \boldsymbol{\theta}\right)\nabla_{\boldsymbol{\theta}} u\left(x_0, \boldsymbol{\theta}\right) + u\left(x_N, \boldsymbol{\theta}\right)\nabla_{\boldsymbol{\theta}} u\left(x_N, \boldsymbol{\theta}\right)\right] \\ =& -\frac{1}{N}\sum_{j=1}^{N-1} K\left(x_i, x_j\right)\left(\left(A_N^2 u\right)_j - \left(A_N g\right)_j\right) \\ &- K\left(x_i, x_0\right) u\left(x_0\right) - K\left(x_i, x_N\right) u\left(x_N\right),\end{aligned}$$

where $K := \left(K\left(x_i, x_j\right)\right)_{(N-1)\times(N-1)} = \left(\nabla_{\boldsymbol{\theta}} u\left(x_i, \boldsymbol{\theta}\right) \cdot \nabla_{\boldsymbol{\theta}} u\left(x_j, \boldsymbol{\theta}\right)\right)_{(N-1)\times(N-1)}$ is the spectrum defined in Sect. 3.2.1.

We similarly define the error as $e := u - u^*$. Then,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}e\left(x_i, \boldsymbol{\theta}\right) =& -\frac{1}{N}\left[\sum_{i=1}^{N-1} K\left(x_i, x_j\right)\left(A_N^2 e\right)_j\right. \\ &\left. -NK\left(x_i, x_0\right) e_0 - NK\left(x_i, x_N\right) e_N\right],\end{aligned}$$

where $K = \left(K\left(x_i, x_j\right)\right), i, j = 0, \cdots, N$. Define $\bar{e} = (e_0, \cdots, e_N)$ and consider the augmented matrix:

$$\bar{A}_N \bar{\boldsymbol{v}} = \frac{1}{h^2}\begin{pmatrix} C & & & & & \\ & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & \ddots & \ddots & \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 2 \\ & & & & & C \end{pmatrix}\begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_N \end{pmatrix},$$

where C is to be determined. For $j = 1, \cdots, N-1$, $\bar{A}_N^2 \bar{\boldsymbol{e}}\big|_j = A_N^2 \boldsymbol{e}\big|_j$, for $j = 0$,

$$\begin{aligned}\bar{A}_N^2 \bar{e}\big|_0 &= \left(\frac{C}{h^2}\right)^2 e_0 = N e_0 \Leftrightarrow \frac{C}{h^2} = \sqrt{N}, \\ C &= h^2 \frac{1}{\sqrt{h}} = h^{3/2}.\end{aligned}$$

Taking together, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{e} = -\frac{1}{N} K \bar{A}_N^2 \bar{e}. \tag{59}$$

Although for the differential operator (derivative w.r.t. input), the lower frequency mode converges more slowly, for the integral operator (loss consists of the summation w.r.t. input),

i.e., spectrum K, the lower frequency mode (see Sect. 3.2) converges faster. Therefore, there is a competition between K and A_N, which is the competition between integral and differential operators (also see analysis in E et al. [33]). The effect of the neural network can also be understood as a preconditioner. Another easy way to understand why the differential operator enables faster convergence of the high frequency is that the Fourier transform of $\nabla u(x)$ is $\xi \hat{u}(\xi)$, that is, a higher frequency would have a higher weight in the loss function.

5.4 Algorithm Design to Overcome the Challenge of High Frequency

The F-Principle provides valuable theoretical insights of the limit of DNN-based algorithms, that is, the challenge of the high-frequency [114].

To overcome the challenge of the high frequency in DNN-based algorithms, a series of methods are proposed. Some approaches are reviewed as follows.

Learning Fourier coefficients PhaseDNN [15] converts the high-frequency component of the data downward to a low-frequency spectrum for learning, and then converts the learned one back to the original high frequency. Another way to understand PhaseDNN is to expand the target function by a Fourier series, and neural networks are used to learn the coefficients. Peng et al. [89] call such a method as Prior Dictionary-based Physics-Informed Neural Networks (PD-PINNs). However, due to the fact that number of Fourier terms exponentially increases with the dimension, the PhaseDNN would suffer from the curse of dimensionality.

Multi-scale DNN To alleviate the high-frequency difficulty for the high-dimensional problem, a Multi-scale DNN (MscaleDNN) method, originally proposed in Cai and Xu [15] and completed in Liu et al. [73], considers the frequency conversion only in the radial direction. The conversion in the frequency space can be done by a scaling, which is equivalent to an inverse scaling in the spatial space. Therefore, we can use the following ansatz to fit high-frequency data:

$$f(\boldsymbol{x}) \sim \sum_{i=1}^{M} f_{\theta^{n_i}}(\alpha_i \boldsymbol{x}). \tag{60}$$

This can be easily implemented by multiplying the input to different neurons in the first hidden layer with different constant scalings. Figure 15 shows two examples of MscaleDNN structures.

MscaleDNN-1 For the first kind, a MscaleDNN takes the following form:

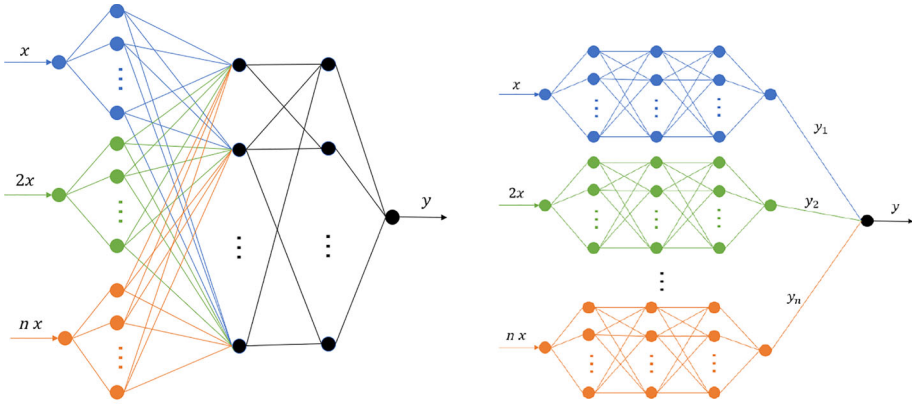
$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}^{[L-1]}\sigma \circ (\cdots (\boldsymbol{W}^{[1]}\sigma \circ (\boldsymbol{K} \odot (\boldsymbol{W}^{[0]}\boldsymbol{x}) + \boldsymbol{b}^{[0]}) + \boldsymbol{b}^{[1]}) \cdots) + \boldsymbol{b}^{[L-1]}, \tag{61}$$

where $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{W}^{[l]} \in \mathbb{R}^{m_{l+1} \times m_l}$, m_l is the neuron number of the l-th hidden layer, $m_0 = d$, $\boldsymbol{b}^{[l]} \in \mathbb{R}^{m_{l+1}}$, σ is a scalar function and "$\circ$" means entry-wise operation, $\odot$ is the Hadamard product and

$$\boldsymbol{K} = (\underbrace{a_1, a_1, \cdots, a_1}_{\text{1st part}}, a_2, \cdots, a_{i-1}, \underbrace{a_i, a_i, \cdots, a_i}_{i\text{th part}}, \cdots, \underbrace{a_N, a_N \cdots, a_N}_{N\text{th part}})^{\mathrm{T}}, \tag{62}$$

where $\boldsymbol{K} \in \mathbb{R}^{m_1}$, $a_i = i$ or $a_i = 2^{i-1}$. This structure is called Multi-scale DNN-1 (MscaleDNN-1).

MscaleDNN-2 A second kind of multi-scale DNN is given in Fig. 15b, as a sum of N subnetworks, in which each scale input goes through a subnetwork. In MscaleDNN-2, weight matrices from $W^{[1]}$ to $W^{[L-1]}$ are block diagonal. Again, the scale coefficient $a_i = i$ or $a_i = 2^{i-1}$.



(a) MscaleDNN-1 (b) MscaleDNN-2

Fig. 15 Two MscaleDNN structures. Reprinted from Liu et al. [73]

Wang et al. [107] and Li et al. [67, 68] further use MscaleDNN to solve more multi-scale problems. Another key factor in practical experiments, without too much understanding, is the effect of the activation function. Liu et al. [73] use activation functions with compact support. Huang et al. [53] and Li et al. [68] adopt the MscaleDNN structure and use the sine and cosine functions as the activation function with different scalings for neurons in the first hidden layer, obtaining better results in solving PDEs.

Fourier feature network Tancik et al. [106] map input $\boldsymbol{x}$ to

$$\gamma(\boldsymbol{x}) = [a_1 \cos(2\pi \boldsymbol{b}_1^{\mathrm{T}}\boldsymbol{x}), a_1 \cos(2\pi \boldsymbol{b}_1^{\mathrm{T}}\boldsymbol{x}), \cdots, a_m \cos(2\pi \boldsymbol{b}_m^{\mathrm{T}}\boldsymbol{x}), a_m \cos(2\pi \boldsymbol{b}_m^{\mathrm{T}}\boldsymbol{x})]$$

for imaging reconstruction tasks. $\gamma(\boldsymbol{x})$ is then used as the input to the neural network. Wang et al. [109] extend the Fourier feature network for the PDE problem, where the selection for b_i is from different ranges. Mildenhall et al. [85] successfully apply this multi-scale Fourier feature input in the neural radiance fields for view synthesis.

Adaptive activation functions Jagtap et al. [55] replace the activation function $\sigma(x)$ by a $\sigma(\mu a x)$, where μ is a fixed scale factor with $\mu \geqslant 1$ and a is a trainable variable shared for all neurons. Liang et al. [71] employ several basic functions and their learnable linear combination to construct neuron-wise data-driven activation functions.

Large weight for high frequency Biland et al. [9] explicitly impose high frequencies with the higher priority in the loss function to accelerate the simulation of fluid dynamics. However, the Fourier transform for the high-dimensional function is computational costly.

6 Anti-F-Principle

The F-Principle is rather common in training DNNs. As we have understood the F-Principle to a certain extent, it is also easy to construct examples in which the F-Principle does not hold, i.e., the anti-F-Principle. As analyzed in Sect. 3.2.3, if the priority of the high frequency is too high, the optimization problem would lead to trivial solutions. Examples in the over-parameterized finite-element method in Fig. 14 is also an example. In this section, we review some anti-F-Principle examples.

6.1 Derivative w. r. t. Input

Imposing high priority on high frequency can alleviate the effect of the F-Principle and sometimes a phenomenon that follows the anti-F-Principle can be observed. Similar to Sect. 5.3, if the loss function contains the gradient of the DNN output w.r.t. the input, it is equivalent to impose higher frequency with higher weight in the loss function. Then, whether there exists an F-Principle depends on the competition between the activation regularity and the loss function. If the loss function endorses more priority for the high frequency to compensate for the low-priority induced by the activation function, an anti-F-Principle emerges. Since gradient often exists in solving PDEs, the anti-F-Principle can hold in solving a PDE by designing a loss with high-order derivatives. Some analysis and numerical experiments can also be found in Lu et al. [75] and E et al. [33].

6.2 Large Weights

Another way to observe the phenomenon that follows the anti-F-Principle is using large values for network weights. As shown in the analysis of the ideal setting in Sect. 3.1, large weights alleviate the dominance of low-frequency in (9). In addition, large values would also cause large fluctuation of DNN output at initialization (experiments can be seen in Xu et al. [115]), the amplitude term in (9) may endorse high frequency larger priority, leading to an anti-F-Principle, which is also studied by Yang and Salman [117]. In the NTK regime [54], Zhang et al. [127] theoretically show that the fluctuation of the initial output would be kept in the learned function after training.

7 Conclusion

The F-Principle is very general and important for training DNNs. It serves as a basic principle to understand DNNs and inspires the design of DNNs. As a good starting point, the F-Principle leads to more interesting studies for a better understanding of DNNs. For example, the empirical study also finds the F-Principle holds in the non-gradient training process [81]. It remains unclear how to build a theory of the F-Principle for general DNNs with arbitrary sample distribution and how to study the generalization error. The precise description of the F-Principle is only done in the NTK regime [54]. It is not clear whether it is possible to obtain a similarly precise description in the mean-field regime described by PDEs [82, 94, 103]. The Fourier analysis can be used to study DNNs from other perspectives, such as the effect of different image frequencies on the learning results.

As a general implicit bias, the F-Principle is insufficient to characterize the exact details of the training process of DNNs beyond NTK. To study the nonlinear behavior of DNNs in detail, it is important to study DNNs from other perspectives, such as the loss landscape, the effect of width and depth, the effect of initialization, etc. For example, Zhang et al. [127] and Luo et al. [79] have studied how initialization affects the implicit bias of DNNs and Luo et al. [79] draw a phase diagram for wide two-layer ReLU DNNs [79]. Zhang et al. [124, 128] show an embedding principle that the loss landscape of a DNN "contains" all the critical points of all the narrower DNNs.

Acknowledgements This work is sponsored by the National Key R&D Program of China Grant No. 2022YFA1008200 (Z. X., Y. Z., T. L.), the National Natural Science Foundation of China Grant Nos. 92270001 (Z. X.), 12371511 (Z. X.), 12101402 (Y. Z.), 12101401 (T. L.), the Lingang Laboratory Grant No. LG-QS-

202202-08 (Y. Z.), the Shanghai Municipal Science and Technology Key Project No. 22JC1401500 (T. L.), the Shanghai Municipal of Science and Technology Major Project No. 2021SHZDZX0102, and the HPC of School of Mathematical Sciences and the Student Innovation Center, and the Siyuan-1 cluster supported by the Center for High Performance Computing at Shanghai Jiao Tong University.

Compliance with Ethical Standards

Conflict of Interest The authors have no conflicts of interest to declare.

References

1. Agarwal, R., Frosst, N., Zhang, X., Caruana, R., Hinton, G.E.: Neural additive models: interpretable machine learning with neural nets. arXiv:2004.13912 (2020)
2. Arora, S., Du, S., Hu, W., Li, Z., Wang, R.: Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In: International Conference on Machine Learning, pp. 322–332 (2019)
3. Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., Lacoste-Julien, S.: A closer look at memorization in deep networks. In: International Conference on Machine Learning, pp. 233–242 (2017)
4. Aubin, B., Maillard, A., Barbier, J., Krzakala, F., Macris, N., Zdeborová, L.: The committee machine: computational to statistical gaps in learning a two-layers neural network. Adv. Neural Inf. Process. Syst. **31**, 3223–3234 (2018)
5. Baratin, A., George, T., Laurent, C., Hjelm, R.D., Lajoie, G., Vincent, P., Lacoste-Julien, S.: Implicit regularization via neural feature alignment. arXiv:2008.00938 (2020)
6. Basri, R., Galun, M., Geifman, A., Jacobs, D., Kasten, Y., Kritchman, S.: Frequency bias in neural networks for input of non-uniform density. In: International Conference on Machine Learning, pp. 685–694 (2020)
7. Basri, R., Jacobs, D., Kasten, Y., Kritchman, S.: The convergence rate of neural networks for learned functions of different frequencies. Adv. Neural Inf. Process. Syst. **32**, 4761–4771 (2019)
8. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition. arXiv:2008.03824 (2020)
9. Biland, S., Azevedo, V.C., Kim, B., Solenthaler, B.: Frequency-aware reconstruction of fluid simulations with generative networks. arXiv:1912.08776 (2019)
10. Bordelon, B., Canatar, A., Pehlevan, C.: Spectrum dependent learning curves in kernel regression and wide neural networks. In: International Conference on Machine Learning, pp. 1024–1034 (2020)
11. Breiman, L.: Reflections after refereeing papers for nips. In: The Mathematics of Generalization, pp. 11–15 (1995)
12. Brenner, S.C., Scott, L.R.: The Mathematical Theory of Finite Element Methods. Springer, New York (2008)
13. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv:2005.14165 (2020)
14. Cai, W., Li, X., Liu, L.: A phase shift deep neural network for high frequency approximation and wave problems. SIAM J. Sci. Comput. **42**(5), 3285–3312 (2020)
15. Cai, W., Xu, Z.-Q.J.: Multi-scale deep neural networks for solving high dimensional PDEs. arXiv:1910.11710 (2019)
16. Campo, M., Chen, Z., Kung, L., Virochsiri, K., Wang, J.: Band-limited soft actor critic model. arXiv:2006.11431 (2020)
17. Camuto, A., Willetts, M., Şimşekli, U., Roberts, S., Holmes, C.: Explicit regularisation in Gaussian noise injections. arXiv:2007.07368 (2020)
18. Cao, Y., Fang, Z., Wu, Y., Zhou, D.-X., Gu, Q.: Towards understanding the spectral bias of deep learning. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, pp. 2205–2211 (2021)
19. Chakrabarty, P.: The spectral bias of the deep image prior. In: Bayesian Deep Learning Workshop and Advances in Neural Information Processing Systems (NeurIPS) (2019)
20. Chen, G.-Y., Gan, M., Chen, C.P., Zhu, H.-T., Chen, L.: Frequency principle in broad learning system. IEEE Trans. Neural Netw. Learn. Syst. **33**, 6983 (2021)

21. Chen, H., Lin, M., Sun, X., Qi, Q., Li, H., Jin, R.: MuffNet: multi-layer feature federation for mobile deep learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (2019)
22. Chen, Y., Li, G., Jin, C., Liu, S., Li, T.: SSD-GAN: measuring the realness in the spatial and spectral domains. Proc. AAAI Conf. Artif. Intell. **35**, 1105–1112 (2021)
23. Chizat, L., Bach, F.: On the global convergence of gradient descent for over-parameterized models using optimal transport. Adv. Neural Inf. Process. Syst. **31**, 3036–3046 (2018)
24. Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y.: The loss surfaces of multilayer networks. In: Artificial Intelligence and Statistics, pp. 192–204 (2015)
25. Deng, X., Zhang, Z.M.: Is the meta-learning idea able to improve the generalization of deep neural networks on the standard supervised learning? In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 150–157 (2021)
26. Dissanayake, M., Phan-Thien, N.: Neural-network-based approximations for solving partial differential equations. Commun. Numer. Methods Eng. **10**(3), 195–201 (1994)
27. Dong, B., Hou, J., Lu, Y., Zhang, Z.: Distillation $\approx$ early stopping? Harvesting dark knowledge utilizing anisotropic information retrieval for overparameterized neural network. arXiv:1910.01255 (2019)
28. Dyson, F.: A meeting with Enrico Fermi. Nature **427**(6972), 297 (2004)
29. E, W., Han, J., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**(4), 349–380 (2017)
30. E, W., Han, J., Jentzen, A.: Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning. Nonlinearity **35**(1), 278 (2021)
31. E, W., Ma, C., Wang, J.: Model reduction with memory and the machine learning of dynamical systems. Commun. Comput. Phys. **25**(4), 947–962 (2018)
32. E, W., Ma, C., Wu, L.: A priori estimates of the population risk for two-layer neural networks. Commun. Math. Sci. **17**(5), 1407–1425 (2019)
33. E, W., Ma, C., Wu, L.: Machine learning from a continuous viewpoint, I. Sci. China Math. **63**, 2233–2266 (2020)
34. E, W., Yu, B.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**(1), 1–12 (2018)
35. Engel, A., Broeck, C.V.d.: Statistical Mechanics of Learning. Cambridge University Press, Cambridge (2001)
36. Evans, L.C.: Partial Differential Equations. American Mathematical Society, Providence, Rhode Island (2010)
37. Fan, Y., Lin, L., Ying, L., Zepeda-Núnez, L.: A multiscale neural network based on hierarchical matrices. Multiscale Model. Simul. **17**(4), 1189–1213 (2019)
38. Frankle, J., Carbin, M.: The lottery ticket hypothesis: finding sparse, trainable neural networks. arXiv:1803.03635 (2018)
39. Fu, Y., Guo, H., Li, M., Yang, X., Ding, Y., Chandra, V., Lin, Y.: CPT: efficient deep neural network training via cyclic precision. arXiv:2101.09868 (2021)
40. Fu, Y., You, H., Zhao, Y., Wang, Y., Li, C., Gopalakrishnan, K., Wang, Z., Lin, Y.: Fractrain: fractionally squeezing bit savings both temporally and spatially for efficient DNN training. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, Virtual (2020)
41. Giryes, R., Bruna, J.: How can we use tools from signal processing to understand better neural networks? Inside Signal Processing Newsletter (2020)
42. Goldt, S., Mézard, M., Krzakala, F., Zdeborová, L.: Modeling the influence of data structure on learning in neural networks: the hidden manifold model. Phys. Rev. X **10**(4), 041044 (2020)
43. Guo, M., Fathi, A., Wu, J., Funkhouser, T.: Object-centric neural scene rendering. arXiv:2012.08503 (2020)
44. Han, J., Ma, C., Ma, Z., Weinan, E.: Uniformly accurate machine learning-based hydrodynamic models for kinetic equations. Proc. Natl. Acad. Sci. **116**(44), 21983–21991 (2019)
45. Han, J., Zhang, L., Car, R., E, W.: Deep potential: a general representation of a many-body potential energy surface. Commun. Comput. Phys. **23**, 3 (2018)
46. Häni, N., Engin, S., Chao, J.-J., Isler, V.: Continuous object representation networks: novel view synthesis without target view supervision. arXiv:2007.15627 (2020)
47. He, J., Li, L., Xu, J., Zheng, C.: ReLU deep neural networks and linear finite elements. arXiv:1807.03973 (2018)
48. He, J., Xu, J.: MgNet: A unified framework of multigrid and convolutional neural network. Sci. China Math. **62**(7), 1331–1354 (2019)

49. He, S., Wang, X., Shi, S., Lyu, M.R., Tu, Z.: Assessing the bilingual knowledge learned by neural machine translation models. arXiv:2004.13270 (2020)
50. Hennigh, O., Narasimhan, S., Nabian, M.A., Subramaniam, A., Tangsali, K., Fang, Z., Rietmann, M., Byeon, W., Choudhry, S.: NVIDIA SimNet: an AI-accelerated multi-physics simulation framework. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M. (eds) Computational Science-ICCS 2021. Lecture Notes in Computer Science, vol. 12746, pp. 447–461. Springer, Cham (2021)
51. Hu, W., Xiao, L., Adlam, B., Pennington, J.: The surprising simplicity of the early-time learning dynamics of neural networks. arXiv:2006.14599 (2020)
52. Huang, J., Wang, H., Yang, H.: Int-Deep: a deep learning initialized iterative method for nonlinear problems. J. Comput. Phys. **419**, 109675 (2020)
53. Huang, X., Liu, H., Shi, B., Wang, Z., Yang, K., Li, Y., Weng, B., Wang, M., Chu, H., Zhou, J., Yu, F., Hua, B., Chen, L., Dong, B.: Solving partial differential equations with point source based on physics-informed neural networks. arXiv:2111.01394 (2021)
54. Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: convergence and generalization in neural networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 8580–8589 (2018)
55. Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E.: Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J. Comput. Phys. **404**, 109136 (2020)
56. Jiang, L., Dai, B., Wu, W., Loy, C.C.: Focal frequency loss for generative models. arXiv:2012.12821 (2020)
57. Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., Bengio, S.: Fantastic generalization measures and where to find them. In: International Conference on Learning Representations (2019)
58. Jin, H., Montúfar, G.: Implicit bias of gradient descent for mean squared error regression with wide neural networks. arXiv:2006.07356 (2020)
59. Jin, P., Lu, L., Tang, Y., Karniadakis, G.E.: Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness. Neural Netw. **130**, 85–99 (2020)
60. Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., Zhang, H.: SGD on neural networks learns functions of increasing complexity. Adv. Neural Inf. Process. Syst. **32**, 3496–3506 (2019)
61. Khoo, Y., Ying, L.: SwitchNet: a neural network model for forward and inverse scattering problems. SIAM J. Sci. Comput. **41**(5), 3182–3201 (2019)
62. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
63. Kopitkov, D., Indelman, V.: Neural spectrum alignment: empirical study. In: International Conference on Artificial Neural Networks, pp. 168–179. Springer (2020)
64. Lampinen, A.K., Ganguli, S.: An analytic theory of generalization dynamics and transfer learning in deep linear networks. In: The International Conference on Learning Representations (2019)
65. Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., Pennington, J.: Wide neural networks of any depth evolve as linear models under gradient descent. Adv. Neural Inf. Process. Syst. **32**, 8572–8583 (2019)
66. Li, M., Soltanolkotabi, M., Oymak, S.: Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In: International Conference on Artificial Intelligence and Statistics, pp. 4313–4324 (2020)
67. Li, X.-A., Xu, Z.-Q.J., Zhang, L.: A multi-scale DNN algorithm for nonlinear elliptic equations with multiple scales. Commun. Comput. Phys. **28**(5), 1886–1906 (2020). https://doi.org/10.4208/cicp.OA-2020-0187
68. Li, X.-A., Xu, Z.-Q.J., Zhang, L.: Subspace decomposition based DNN algorithm for elliptic-type multi-scale PDEs. J. Comput. Phys. **488**, 112242 (2023). https://doi.org/10.2139/ssrn.4020731
69. Li, Y., Peng, W., Tang, K., Fang, M.: Spatio-frequency decoupled weak-supervision for face reconstruction. Comput. Intell. Neurosci. **2022**, 1–12 (2022)
70. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv:2010.08895 (2020)
71. Liang, S., Lyu, L., Wang, C., Yang, H.: Reproducing activation function for deep learning. arXiv:2101.04844 (2021)
72. Lin, J., Camoriano, R., Rosasco, L.: Generalization properties and implicit regularization for multiple passes SGM. In: International Conference on Machine Learning, pp. 2340–2348 (2016)
73. Liu, Z., Cai, W., Xu, Z.-Q.J.: Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains. Commun. Comput. Phys. **28**(5), 1970–2001 (2020)
74. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat. Mach. Intell. **3**(3), 218–229 (2021)

75. Lu, L., Meng, X., Mao, Z., Karniadakis, G.E.: DeepXDE: a deep learning library for solving differential equations. SIAM Rev. **63**(1), 208–228 (2021)
76. Luo, T., Ma, Z., Wang, Z., Xu, Z.-Q.J., Zhang, Y.: Fourier-domain variational formulation and its well-posedness for supervised learning. arXiv:2012.03238 (2020)
77. Luo, T., Ma, Z., Xu, Z.-Q.J., Zhang, Y.: On the exact computation of linear frequency principle dynamics and its generalization. arXiv:2010.08153 (2020)
78. Luo, T., Ma, Z., Xu, Z.-Q.J., Zhang, Y.: Theory of the frequency principle for general deep neural networks. CSIAM Trans. Appl. Math. **2**(3), 484–507 (2021). https://doi.org/10.4208/csiam-am.SO-2020-0005
79. Luo, T., Xu, Z.-Q.J., Ma, Z., Zhang, Y.: Phase diagram for two-layer ReLU neural networks at infinite-width limit. J. Mach. Learn. Res. **22**, 1–47 (2021)
80. Ma, C., Wu, L., E., W.: The slow deterioration of the generalization error of the random feature model. In: Mathematical and Scientific Machine Learning, pp. 373–389 (2020)
81. Ma, Y., Xu, Z.-Q.J., Zhang, J.: Frequency principle in deep learning beyond gradient-descent-based training. arXiv:2101.00747 (2021)
82. Mei, S., Montanari, A., Nguyen, P.-M.: A mean field view of the landscape of two-layer neural networks. Proc. Natl. Acad. Sci. **115**(33), 7665–7671 (2018)
83. Michoski, C., Milosavljevic, M., Oliver, T., Hatch, D.: Solving irregular and data-enriched differential equations using deep neural networks. arXiv:1905.04351 (2019)
84. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision, pp. 405–421. Springer (2020)
85. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**(1), 99–106 (2021)
86. Mingard, C., Skalse, J., Valle-Pérez, G., Martínez-Rubio, D., Mikulik, V., Louis, A.A.: Neural networks are a priori biased towards boolean functions with low entropy. arXiv:1909.11522 (2019)
87. Nye, M., Saxe, A.: Are efficient deep representations learnable? arXiv:1807.06399 (2018)
88. Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., Zhou, X.: Neural body: implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9054–9063 (2021)
89. Peng, W., Zhou, W., Zhang, J., Yao, W.: Accelerating physics-informed neural network training with prior dictionaries. arXiv:2004.08151 (2020)
90. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-NeRF: neural radiance fields for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10318–10327 (2021)
91. Rabinowitz, N.C.: Meta-learners' learning dynamics are unlike learners'. arXiv:1905.01320 (2019)
92. Rahaman, N., Arpit, D., Baratin, A., Draxler, F., Lin, M., Hamprecht, F.A., Bengio, Y., Courville, A.: On the spectral bias of deep neural networks. In: International Conference on Machine Learning (2019)
93. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
94. Rotskoff, G.M., Vanden-Eijnden, E.: Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 7146–7155 (2018)
95. Saxe, A.M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B.D., Cox, D.D.: On the information bottleneck theory of deep learning. J. Stat. Mech. Theory Exp. **2019**(12), 124020 (2019). https://doi.org/10.1088/1742-5468/ab3985
96. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In: The International Conference on Learning Representations (2014)
97. Schwarz, K., Liao, Y., Geiger, A.: On the frequency bias of generative models. Adv. Neural Inf. Process. Syst. **34**, 18126 (2021)
98. Shalev-Shwartz, S., Shamir, O., Shammah, S.: Failures of gradient-based deep learning. In: International Conference on Machine Learning, pp. 3067–3075 (2017)
99. Sharma, R., Ross, A.: D-NetPAD: an explainable and interpretable iris presentation attack detector. In: 2020 IEEE International Joint Conference on Biometrics (IJCB), pp. 1–10 (2020)
100. Shen, J., Tang, T., Wang, L.L.: Spectral Methods: Algorithm, Analysis and Applications. Springer, Berlin (2011)
101. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv:1703.00810 (2017)

102. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings (2015)
103. Sirignano, J., Spiliopoulos, K.: Mean field analysis of neural networks: a central limit theorem. Stoch. Process. Appl. **130**(3), 1820–1852 (2020)
104. Strofer, C.M., Wu, J.-L., Xiao, H., Paterson, E.: Data-driven, physics-based feature extraction from fluid flow fields using convolutional neural networks. Commun. Comput. Phys. **25**(3), 625–650 (2019)
105. Tancik, M., Mildenhall, B., Wang, T., Schmidt, D., Srinivasan, P.P., Barron, J.T., Ng, R.: Learned initializations for optimizing coordinate-based neural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2846–2855 (2021)
106. Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. Adv. Neural Inf. Process. Syst. **33**, 7537–7547 (2020)
107. Wang, B., Zhang, W., Cai, W.: Multi-scale deep neural network (MscaleDNN) methods for oscillatory stokes flows in complex domains. Commun. Comput. Phys. **28**(5), 2139–2157 (2020)
108. Wang, J., Xu, Z.-Q.J., Zhang, J., Zhang, Y.: Implicit bias with Ritz-Galerkin method in understanding deep learning for solving PDEs. arXiv:2002.07989 (2020)
109. Wang, S., Wang, H., Perdikaris, P.: On the eigenvector bias of Fourier feature networks: from regression to solving multi-scale PDEs with physics-informed neural networks. Comput. Methods Appl. Mech. Eng. **384**, 113938 (2021)
110. Xi, Y., Jia, W., Zheng, J., Fan, X., Xie, Y., Ren, J., He, X.: DRL-GAN: dual-stream representation learning GAN for low-resolution image classification in UAV applications. IEEE J. Select. Top. Appl. Earth Observ. Remote Sens. **14**, 1705–1716 (2020)
111. Xie, B., Liang, Y., Song, L.: Diverse neural network learns true target functions. Int. Conf. Artif. Intell. Stat. **54**, 1216–1224 (2017)
112. Xu, R., Wang, X., Chen, K., Zhou, B., Loy, C.C.: Positional encoding as spatial inductive bias in GANs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13569–13578 (2021)
113. Xu, Z.-Q.J.: Understanding training and generalization in deep learning by Fourier analysis. arXiv:1808.04295 (2018)
114. Xu, Z.-Q.J., Zhang, Y., Luo, T., Xiao, Y., Ma, Z.: Frequency principle: Fourier analysis sheds light on deep neural networks. Commun. Comput. Phys. **28**(5), 1746–1767 (2020)
115. Xu, Z.-Q.J., Zhang, Y., Xiao, Y.: Training behavior of deep neural network in frequency domain. In: International Conference on Neural Information Processing, pp. 264–274. Springer (2019)
116. Xu, Z.-Q.J., Zhou, H.: Deep frequency principle towards understanding why deeper learning is faster. Proc. AAAI Conf. Artif. Intell. **35**, 10541 (2021)
117. Yang, G., Salman, H.: A fine-grained spectral perspective on neural networks. arXiv:1907.10599 (2019)
118. Yang, M., Wang, Z., Chi, Z., Zhang, Y.: FreGAN: exploiting frequency components for training GANs under limited data. arXiv:2210.05461 (2022)
119. You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Lin, Y., Wang, Z., Baraniuk, R.G.: Drawing early-bird tickets: towards more efficient training of deep networks. In: International Conference on Learning Representations (2020)
120. Zang, Y., Bao, G., Ye, X., Zhou, H.: Weak adversarial networks for high-dimensional partial differential equations. J. Comput. Phys. **411**, 109409 (2020)
121. Zdeborová, L.: Understanding deep learning is also a job for physicists. Nat. Phys. **16**, 1–3 (2020). https://doi.org/10.1038/s41567-020-0929-2
122. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: 5th International Conference on Learning Representations (2017)
123. Zhang, L., Luo, T., Zhang, Y., Xu, Z.-Q.J., Ma, Z.: MOD-NET: a machine learning approach via model-operator-data network for solving PDEs. arXiv:2107.03673 (2021)
124. Zhang, Y., Li, Y., Zhang, Z., Luo, T., Xu, Z.-Q.J.: Embedding principle: a hierarchical structure of loss landscape of deep neural networks. arXiv:2111.15527 (2021)
125. Zhang, Y., Luo, T., Ma, Z., Xu, Z.-Q.J.: A linear frequency principle model to understand the absence of overfitting in neural networks. Chin. Phys. Lett. **38**(3), 038701 (2021)
126. Zhang, Y., Xu, Z.-Q.J., Luo, T., Ma, Z.: Explicitizing an implicit bias of the frequency principle in two-layer neural networks. arXiv:1905.10264 (2019)
127. Zhang, Y., Xu, Z.-Q.J., Luo, T., Ma, Z.: A type of generalization error induced by initialization in deep neural networks. In: Mathematical and Scientific Machine Learning, pp. 144–164 (2020)
128. Zhang, Y., Zhang, Z., Luo, T., Xu, Z.-Q.J.: Embedding principle of loss landscape of deep neural networks. NeurIPS (2021)

129. Zheng, Q., Babaei, V., Wetzstein, G., Seidel, H.-P., Zwicker, M., Singh, G.: Neural light field 3D printing. ACM Trans. Graph. (TOG) **39**(6), 1–12 (2020)
130. Zhu, H., Qiao, Y., Xu, G., Deng, L., Yu, Y.-F.: DSPNet: a lightweight dilated convolution neural networks for spectral deconvolution with self-paced learning. IEEE Trans. Ind. Inf. **16**, 7392 (2019)